НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2018р.

Магістерська дисертація

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Програмне забезпечення розподілених систем
на тему Клієнт-серверний програмний продукт для забезпечення відеоконференцій

Виконав (-ла): студент (-ка) _6_курсу, групи _ТВ-71мп_
Горбенко Олексій Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник _доц., к.т.н., Третяк В. А._____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент доц., к.т.н., Новаківський Є.В._____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2018

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти другий, магістерський

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В. _____
(прізвище, ініціали) (підпис)
«____» _____ 2018р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Горбенко Олексія Юрійовича
(прізвище, ім’я, по батькові)

1. Тема дисертації Клієнт-серверний програмний продукт для забезпечення відеоконференцій

Науковий керівник Третяк Валерія Анатоліївна, к.т.н., доц.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “___”_____20___року №___

2. Строк подання студентом дисертації _____

3.Об’єкт дослідження технології передачі даних між клієнтськими частинами програмного забезпечення

4. Предмет дослідження технологія WebRTC у якості базисної в розрізі створення програмного продукту для забезпечення відео конференцій

5. Перелік питань, які потрібно розробити: проаналізувати існуючі технології передачі потоків відео та аудіо в розрізі створення відеоконференцій; − удосконалити спосіб створення та управління відеоконференці на серверній частині програмного продукту; − спроектувати архітектуру програмного продукту для забезпечення відеоконференцій з врахуванням запропонованих методів та алгоритмів та − реалізувати серверну та клієнтську частини продукту для забезпечення відеоконференцій

6. Орієнтований перелік ілюстративного матеріалу: Діаграма послідовностей взаємодії користувачів, браузерів та сервера, діаграма класів серверної часттини програмного програмного продукту, діаграма пакетів клієнської частини програмного заюезпечення

7. Орієнтований перелік публікацій: XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів.

8. Дата видачі завдання «30» вересня 2018р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	30.09.2018	
2	Збір інформації	30.10.2018	
3	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі	10.11.2018	
4	Програмна реалізація системи	20.10.2018	
5	Розробка структур окремих підсистем	23.10.2018	
8	Захист програмного продукту	24.10.2018	
9	Передзахист	28.11.2018	
10	Розробка стартап-проекту	01.12.2018	
11	Оформлення пояснювальної записки	16.12.2018	
12	Захист	17.12.2018	

Студент _____ Горбенко О. Ю.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Третяк В. А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Структура й обсяг роботи. Магістерська дисертація складається зі вступу, чотирьох розділів, висновку, переліку посилань з 38 найменувань, 2 додатків і містить 25 рисунків та 22 таблиці. Повний обсяг магістерської дисертації складає 96 сторінок, з яких перелік посилань займає 4 сторінки, додатки – 8 сторінок.

Актуальність теми. Основна увага в роботі присвячена аналізу технологій та технік передачі потоків відео та аудіо для побудови відеоконференцій. Розглядається задача розробки програмного продукту для забезпечення відеоконференцій. Актуальність дослідження існуючих технологій на предмет можливості забезпечення більшої ефективності передачі медіа-потоків обумовлена необхідністю зменшення витрат на підтримку серверної частини програмних систем, що надають можливість проведення віддалених відеоконференцій.

Мета дослідження полягає в аналізі технологій передачі даних та створення програмного продукту для забезпечення відеоконференцій.

Для досягнення поставленої мети були сформульовані наступні **завдання дослідження**, що визначили логіку дослідження та його структуру:

- проаналізувати існуючі технології передачі потоків відео та аудіо в розрізі створення відеоконференцій;
- спроектувати архітектуру програмного продукту для забезпечення відеоконференцій з врахуванням запропонованих методів та алгоритмів;
- реалізувати серверну та клієнтську частини програмного продукту для забезпечення відеоконференцій;
- провести аудит ідеї проекту та методи розроблення маркетингової та ринкової стратегій розвитку програмного продукту.

Об’єктом дослідження є програмне забезпечення технології передачі даних при проведенні відеоконференції.

Предметом дослідження є програмне забезпечення для проведення відеоконференцій з використанням нової технології WebRTC.

Методи дослідження. Для розв'язання поставлених задач використовувалися наступні методи:

- методи і шаблони проектування систем за принципами ООП;
- методи розробки з використанням класифікації за топологічною специфікою.

Наукова новизна отриманих результатів. Найбільш суттєвими науковими результатами магістерської дисертації є:

- вперше реалізовано програмний продукт для забезпечення відеоконференцій з використанням технології WebRTC;

- удосконалено побудову серверної частини програмного продукту за рахунок використання технології WebRTC, що зменшує навантаження на серверну частину програмного продукту для забезпечення відеоконференцій;

- набуло подальшого розвитку способи використання методів побудови клієнтської частини програмного продукту для проведення відеоконференцій з високим рівнем швидкодії.

Практичне значення одержаних результатів роботи полягає в розробці клієнт-серверного програмного продукту для забезпечення відеоконференції у таких областях, як менеджмент віддалених команд розробки, проведення дистанційних лекційних та практичних занять та у якості соціальної мережі.

Практична реалізація одержаних результатів роботи розглянута ТОВ "Інновейшн девелопмент хаб".

Апробація результатів дисертації

Основні положення роботи доповідались і обговорювались на :

Горбенко О.Ю. Веб-ресурс для забезпечення проведення дистанційних лекційних занять / Горбенко О.Ю., Третяк В.А. / Сучасні проблеми наукового забезпечення енергетики: Матеріали XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів, м. Київ, 24–27 квітня 2018 р. У 2 т. – К. : 7 КПІ ім. Ігоря Сікорського», 2018. – Т. 2. – С. 188.

Горбенко О.Ю. Огляд технології WebRTC для реалізації програмного забезпечення відеоконференцій / Горбенко О.Ю., Третяк В.А. / Сталий розвиток – XXI століття: управління, технології, моделі. Дискусії – 2018: колективна монографія / Міненко М.А. та ін., НТУУ КПІ ім. Ігоря Сікорського, Національний університет "Києво-Могилянська академія"; Вища економіко-гуманітарна школа. – Київ, 2018 . – С.467-472.

Ключові слова. *WEBRTC, ВІДЕОКОНФЕРЕНЦІЇ, SERVER-SENT EVENTS, ANGULAR, ДИСТАНЦІЙНЕ*.

ABSTRACT

Structure and scope of work. The master's dissertation consists of an introduction, four sections, a conclusion, a list of references from 40 titles, 3 annexes, and contains 23 figures, 25 tables. The full volume of the master's dissertation is 145 pages, of which the list of links takes 5 pages, applications - 26 pages.

Relevance. The main attention in this work is devoted to the phenomenon of temperature increase in relation to the initial without additional heating, which was found in the process of field experiments and requires further research. In this paper consider, the problem of mathematical simulation of a temperature field in the process of fusion of a bimetal. Consequently, the lack of appropriate simulation tools that would reveal heating in the perforation area, as well as the need to find an adequate mathematical model of the process required for further research, determine the relevance of the topic.

The main attention in this work is devoted to the analysis of technologies and techniques for transmitting video and audio streams for videoconferencing. . In this paper consider the task of developing a software product for videoconferencing. The need to reduce the cost of supporting the server part of the software systems that enable the possibility of remote video conferencing, determines the relevance of the analysis of existing technologies to the possibility of ensuring a greater efficiency of the transfer of media streams.

Goal of the research is to analyze data technology and create a software product fot providing video conferencing between users.

Objectives:

Analyze existing video and audio streaming technologies in the context of videoconferencing.

Design the software architecture for video conferencing, taking into account the proposed methods and algorithms..

Implement the server and client parts of the software to provide video conferencing.Провести аудит ідеї проекту та методи розроблення маркетингової та ринкової стратегій розвитку програмного продукту.

The object of research is the technology of data transmission during video conferencing.

The subject of the research is the use of WebRTC technology to create a client-server software product that provides video conferencing.

Research methods:

- methods for analyzing existing technologies for transferring audio and video streams between users;

- methods and patterns of designing systems according to the principles of the OOP.

Scientific novelty of the obtained results. The most significant scientific results of the master's thesis are:

- for the first time, a software product for the provision of videoconferencing using WebRTC technology has been implemented;

- improved construction of the server part of the software product due to the use of WebRTC technology, which reduces the load on the server part of the software to provide video conferencing;

- has further developed the method of using the methods of constructing a client part of a software product for videoconferencing with a high level of performance.

The practical significance of the results роботи is developing client-server software product to provide video conferencing in areas such as remote command development, remote lecture and practical classes, and as a social network.

The practical realization of the obtained results of work was considered by "Innovation Hub Development".

Approbation of the results of the dissertation

The main provisions of the work were reported and discussed at:

1. XV International scientific and practical conference of postgraduate students, masters, students "Modern problems of scientific support of power engineering" (Kyiv, April 24-27, 2017);

2. Sustainable development - XXI century: management, technology, models. Discussions - 2018: Collective Monograph / Minenko M.A. etc., NTUU KPI them. Igor Sikorsky, National University of Kyiv-Mohyla Academy; Higher economics and humanitarian school. - Kyiv, 2018. - P.467-472.

Keywords. *WEBRTC, ВІДЕОКОНФЕРЕНЦІЇ, SERVER-SENT EVENTS, ANGULAR, REMOTE.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ — програмне забезпечення;

ПП — програмний продукт;

КЛІЄНТ — клієнтська частина програмного продукту;

ICE — протокол налагодження зв'язку RFC 8445;

WebRTC — технологія передачі даних за RFC-7478 та RFC-5245;

ВСТУП

Розвиток швидкісного інтернету розширив межі спілкування між людьми. Відео зв’язок між двома користувачами вже успішно реалізовано в багатьох програмних засобах. Однак в деяких сферах діяльності виникає необхідність забезпечення відео зв’язку між багатьма учасниками. Прикладом може бути забезпечення проведення дистанційних лекційних та практичних занять , проведення засідань в рамках розподілених команд, що часто зустрічається під час розробки програмного забезпечення.

Наразі, технічне забезпечення відео зв’язку між багатьма учасниками не є досконалим. В усіх присутніх на ринку програмних засобах є технічні обмеження по кількості учасників (Skype Desktop – до 25, FaceTime – до 32), і, при збільшенні кількості учасників зменшується якість медіа-потоків (відео та аудіо).

Таким чином, виникає необхідність розробки нового програмного продукту, що дозволяє відео-спілкування між багатьма учасниками (більше 35-ти).

Існуючі програмні продукти, такі як Skype, FaceTime, працюють за принципом клієнт-серверної архітектури. При великому навантаженні сервера якість зв’язку та швидкість її передачі між клієнтами буде спадати зі зростанням активних користувачів. Таким чином, найбільш гостро відчуваються ці недоліки при забезпеченні відео-зв’язку між багатьма користувачами, такими як відеоконференції. Ще більш складною є реалізація систем, що дозволяють одночасне відображення багатьох відео-потоків, що потребується, наприклад, для відеоконференцій в процесі проведення дистанційного навчання або для забезпечення спілкування віддалених команд розробки програмного забезпечення. Саме тому більшість існуючих систем або зовсім не мають можливості створення відеоконференцій, або накладають суттєві обмеження на кількість відео-потоків.

Отже, необхідність зменшення витрат на підтримку серверної частини програмних систем, що надають можливість проведення віддалених відеоконференцій, обумовлює актуальність дослідження існуючих технологій на предмет можливості забезпечення більшої ефективності при передачі медіа-потоків.

Програмний продукт для забезпечення відео конференції повинен підтримувати одночасний зв'язок із багатьма (більше 30-ти) користувачами системи, повинен бути надійним та швидко асинхронно обробляти медіа-потоки відео та аудіо.

Мета дослідження полягає в аналізі технологій передачі даних та створення програмного продукту для забезпечення відео-конференцій.

Для досягнення поставленої мети були сформульовані наступні **завдання дослідження**, що визначили логіку дослідження та його структуру:

1. Проаналізувати існуючі технології передачі потоків відео та аудіо в розрізі підтримки відео-конференцій.
2. Спроектувати архітектуру програмного продукту для забезпечення відео-конференцій з урахуванням запропонованих методів та алгоритмів.
3. Реалізувати серверну та клієнтську частини програмного продукту для забезпечення відео-конференцій.
4. Провести аудит ідеї проекту та методи розроблення маркетингової та ринкової стратегій розвитку програмного продукту.

Об'єктом дослідження є програмне забезпечення технології передачі даних при проведенні відео-конференції.

Предметом дослідження є програмне забезпечення для проведення відеоконференцій з використанням нової технології WebRTC.

Методи дослідження. Для розв'язання поставлених задач використовувалися наступні методи:

- методи і шаблони проектування систем за принципами ООП;
- методи розробки з використанням класифікації за топологічною специфікою.

Наукова новизна одержаних результатів. Найбільш суттєвими науковими результатами магістерської дисертації є:

- вперше реалізовано програмний продукт для забезпечення відео-конференцій з використанням технології WebRTC;

– удосконалено спосіб побудови серверної частини програмного продукту за рахунок використання технології WebRTC, що зменшує навантаження на серверну частину програмного продукту для забезпечення відео-конференцій;

– набуло подальшого розвитку використання методів побудови клієнтської частини програмного продукту для проведення відео-конференцій з високим рівнем швидкодії.

Практичне значення одержаних результатів роботи полягає в програмного продукту для забезпечення відеоконференції з кількістю учасників більше 35-ти та високою якістю медіа-даних у таких областях, як менеджмент віддалених команд розробки та у процесі проведення дистанційних лекційних та практичних занять.

Основні положення роботи доповідались і обговорювались на :

1. Горбенко О.Ю. Веб-ресурс для забезпечення проведення дистанційних лекційних занять / Горбенко О.Ю., Третяк В.А. / Сучасні проблеми наукового забезпечення енергетики: Матеріали XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів, м. Київ, 24–27 квітня 2018 р. У 2 т. – К. : 7 КПІ ім. Ігоря Сікорського», 2018. – Т. 2. – С. 188.

2. Горбенко О.Ю. Огляд технології WebRTC для реалізації програмного забезпечення відеоконференцій / Горбенко О.Ю., Третяк В.А. / Сталий розвиток – XXI століття: управління, технології, моделі. Дискусії – 2018: колективна монографія / Міненко М.А. та ін.. НТУУ КПІ ім. Ігоря Сікорського, Національний університет “Києво-Могилянська академія”; Вища економіко-гуманітарна школа. – Київ, 2018 . – С.467-472.

У першому розділі надається аналіз існуючих технологій, засобів та програмних продуктів для забезпечення відеоконференцій.

У другому розділі проаналізовано та описано обчислювальні методи, засоби, математичні і логічні моделі побудови архітектури програмного продукту для забезпечення відеоконференцій.

Третій розділ присвячено аналізу технік та опису реалізації програмного продукту.

У четвертому розділі представлені методи аналізу доцільності створення програмного продукту, аудит ідеї проекту та методи розроблення маркетингової та ринкової стратегій розвитку програмного продукту.

Результати роботи в частині програмного забезпечення для проведення відеоконференцій впроваджено у виробничий процес ТОВ “ІННОВЕЙШН ДЕВЕЛОПМЕНТ ХАБ”.

1. АНАЛІЗ ТЕХНОЛОГІЙ ПЕРЕДАЧІ МЕДІА-ПОТОКІВ

На сьогоднішній день, в багатьох галузях постає проблема забезпечення якісного відео зв’язку між багатьма учасниками.

Існуючі програмні продукти, такі як Skype, FaceTime та інші, працюють за принципом клієнт-серверної архітектури, де серверна частина програмного продукту стає вузьким місцем. При великому навантаженні сервера, якість зв’язку та швидкість передачі потоків даних між клієнтами спадає зі зростом активних користувачів. Найбільш гостро відчуваються ці недоліки при забезпеченні відео-зв’язку між багатьма користувачами, такими як відеоконференції.

В цьому розділі проводиться порівняльний аналіз існуючих технологій передачі даних між клієнтськими частинами програмних засобів проведення відеоконференцій, різниця архітектури програмних продуктів, що базуються на найбільш розповсюджених протоколах передачі медіа-потоків: HTTP, WebSocket, SSE та WebRTC. Обґрунтовується вибір технології для програмного продукту, що розроблюється.

1.1. Техніки передачі даних Polling/Long Polling та Server-Sent Events

Протокол Hypertext Transfer Protocol (RFC2616) - це протокол концепції запит/відповідь, що визначає наступні об'єкти: клієнти, проксі і сервери.

Клієнт встановлює підключення до сервера з метою надсилання HTTP-запитів. Сервер приймає з'єднання клієнтів, для подальшого обслуговування HTTP-запитів, відправляючи зворотні відповіді. Проксі - це проміжні об'єкти, які можуть бути задіяні в процесі доставки запитів клієнта на сервер та навпаки [1].

Стандартна модель HTTP не передбачає надсилання даних на клієнтську частину обміну без відповідного запиту (не має можливості надсилати дані асинхронно). Таким чином, клієнтська частина веб-програми для імітування асинхронного отримання повідомлень (за стандартною моделлю HTTP), повинна періодично надсилати запити на отримання нових даних.

Однак постійні запити на сервер програмного продукту («short polling» в термінології RFC 6202) можуть зменшувати пропускну здатність, ініціюючи запит/відповідь в обидві сторони навіть за відсутності даних та уповільнення здатності реагування клієнтської частини програмного продукту на дії користувача за рахунок часу очікування клієнтом відповіді від сервера. Чеерез це необхідно знаходити лінію компромісу між швидкістю отримання нових даних та навантаження на серверну частину програмного продукту.

У порівнянні з технікою «short polling», техніка «long polling» дозволяє зменшити навантаження на мережу та обробку сервера за рахунок утримання з'єднання запиту.

За технікою «long polling» клієнтська частина програмного продукту надсилає запит на отримання нових даних, сервер, у свою чергу, надсилає відповідь на запит за умови появи нових даних для відповіді або ж у разі перевищення ліміту часу на з'єднання. Після відповіді сервера клієнт надсилає запит на отримання даних знову.

Життєвий цикл програмного продукту з використанням технології «long polling» можна описати такою послідовністю (рисунок 1.1):

1. Клієнтська частина програмного продукту надсилає запит та переходить у стан очікування відповіді від сервера.
2. Сервер утримує потік виконання запиту до моменту доступності нових даних або тайм-ауту.
3. За надходження нових даних, сервер надсилає відповідь на клієнтський запит з новими даними.
4. Клієнтська частина програмного продукту (за необхідності) очікує завершення «періоду паузи» та надсилає повторний запит на отримання даних.

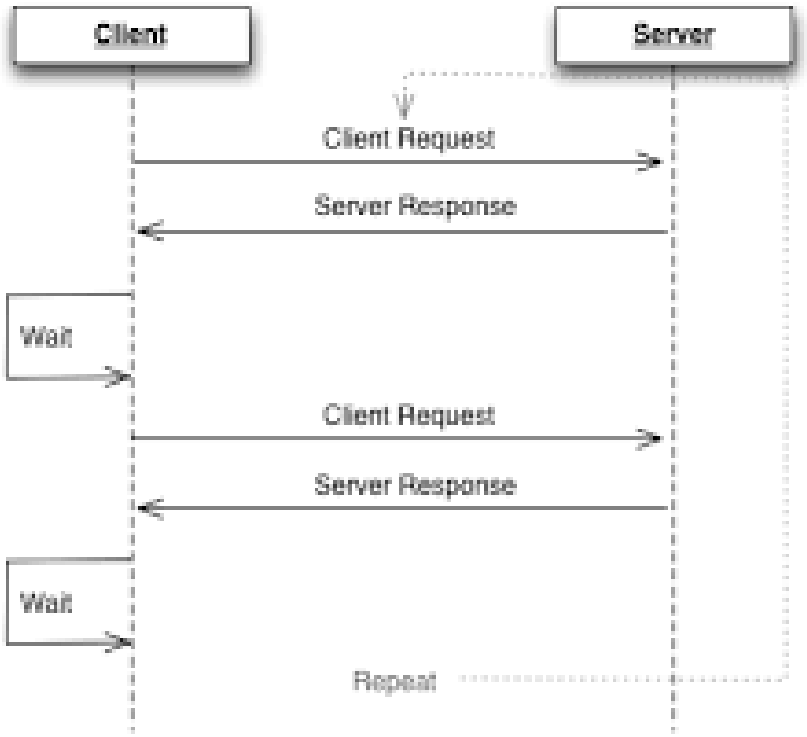

Рисунок 1.1 - Діаграма послідовностей обміну даними за технологією Polling

Основними недоліками використання описаної техніки є [2]:

- Надлишкові заголовки (Header Overhead): за технікою HTTP long polling, кожний запит та відповідь на запит є повноцінним HTTP повідомленням, що містить повний набір HTTP заголовків у фреймі повідомлення. Таким чином, загальний об'єм корисних даних може бути меншим за об'єм заголовків та мета-інформації.
- Затримка (Maximal Latency): після відправлення відповіді на long polling запит, серверна частина програмного продукту повинна очікувати наступного запиту клієнтської частини програмного продукту. За рахунок можливих втрат пакетів від/до клієнта, час очікування може зростати.

Плюсами технік «polling» та «long polling» є використання HTTP протоколу без змін і додаткових налаштувань, кросбраузерність та незалежність від конкретної реалізації налаштувань безпеки обміну даних [3].

До мінусів даних технік можна віднести великі ресурсні затрати мережі, обчислювальних ресурсів клієнта та сервера за рахунок постійних HTTP запитів, обробки даних на серверній частині програмного продукту та відповіді на запит та обробки відповіді. Також за технічної відмови серверної частини програмного продукту, клієнтські частини програмного продукту втрачають можливість до спілкування з іншими клієнтами та сервером[4].

Наступним етапом еволюції передачі подій сервера є створення технік HttpStreaming. Основною концепцією HttpStreaming є налагодження одностороннього з'єднання сервер-клієнт. Такі реалізації, як Bayeux та BOSH побудовані за концепцією змішаного використання технік HTTP Streaming та Polling/Long Polling. Вони вимагають додаткового налаштування проксі-серверів та надлишкової конфігурації розгортки.

Вони діють за концепцією, що базується на утриманні з'єднання за запитом. Наприклад, Bayeux для підтримки двостороннього зв'язку утримує два мережних потоки для забезпечення функціональності двостороннього асинхронного обміну повідомленнями. Один потік – сервер-клієнт, другий – клієнт-сервер. Основною складністю реалізації є налаштування та підтримка серверної частини програмного продукту.

Стандарт Server-Sent Events, у свою чергу, регламентує концепцію надсилання повідомлень із сервера на клієнт. За [5] Server-Sent Events (SSE) є технологією для надсилання повідомлень із сервера до веб-браузера у вигляді DOM-подій.

Основною перевагою над Polling/Long Polling є миттєве надсилання повідомлень із сервера до клієнту (рисунок 1.2).

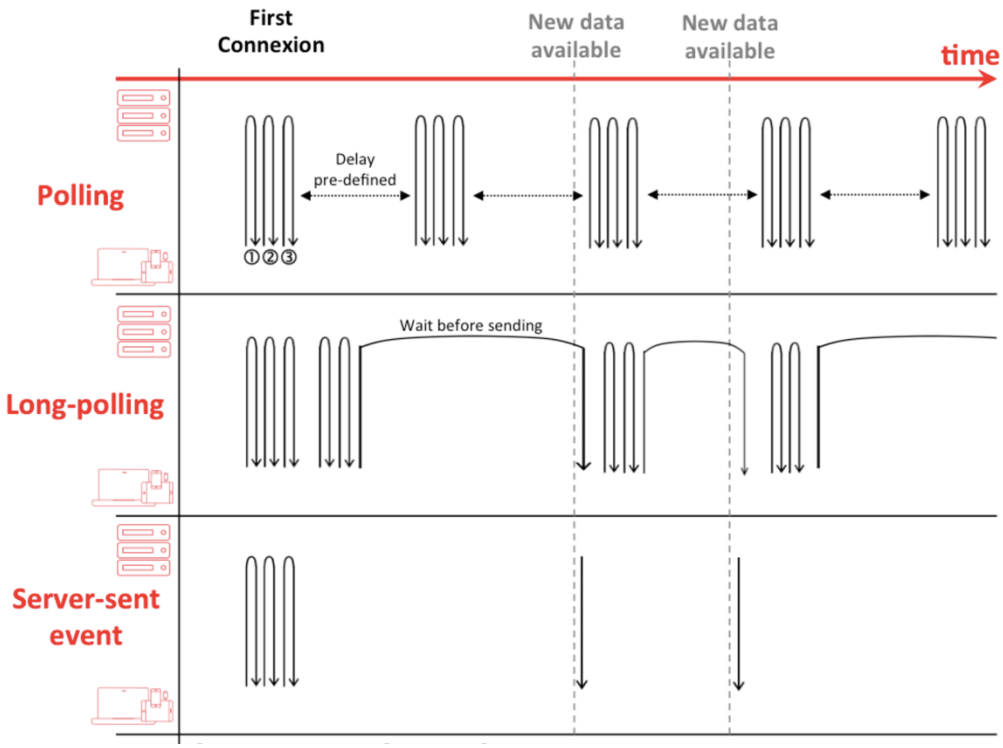

Рисунок 1.2 - Схема порівняння концептів обміну даних Polling/Long Polling та SSE за часом

Він спроектований для оптимізації кросбраузерного мовлення за допомогою JavaScript API під назвою EventSource. Дані закодовані у вигляді типу text/event-stream та передаються на клієнтську частину програмного продукту, використовуючи механізми HTTP Streaming.

Плюсами реалізацій даного стандарту є оптимізоване використання ресурсів сервера, швидкодія прийому повідомлень та простота імплементації серверної та клієнтської частин програмного продукту.

До мінусів реалізацій механізмів HTTP Streaming можна віднести постійне утримання HTTP-з'єднання із сервера та повна залежність від серверної частини програмного продукту.

1.2. Протокол передачі даних WebSocket

Протокол передачі даних WebSocket, що регламентований у RFC-6455 [6], передбачає двосторонній повнодуплексний зв'язок із клієнтською частиною програмного

продукту (серверна і клієнтська частини програмного продукту надсилають і приймають дані з одного потоку) [7].

За RFC 6455, протокол має дві частини: рукостискання - handshake та передачу даних - data transfer (рисунок 1.3 та рисунок 1.4).

З'єднання клієнт-сервер за протоколом WebSocket ініціюється GET запитом на "рукостискання" на WebSocket-сервер [6]:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

Рисунок 1.3 - Запит на з'єднання за проколом WebSocket

За умови підтримки сервера протоколу, сервер згенерує клієнту відповідь про успішне отримання конфігураційних даних та сигнал про початок передачі даних (рисунок 1.4)

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

Рисунок 1.4 - Відповідь сервера на запит з'єднання за протоколом WebSocket

За умови підтримки протоколу браузером, TCP-з'єднання залишається активним. Подальший обмін даних відбувається з використанням надсилання WebSocket-фреймів (рисунок 1.5).

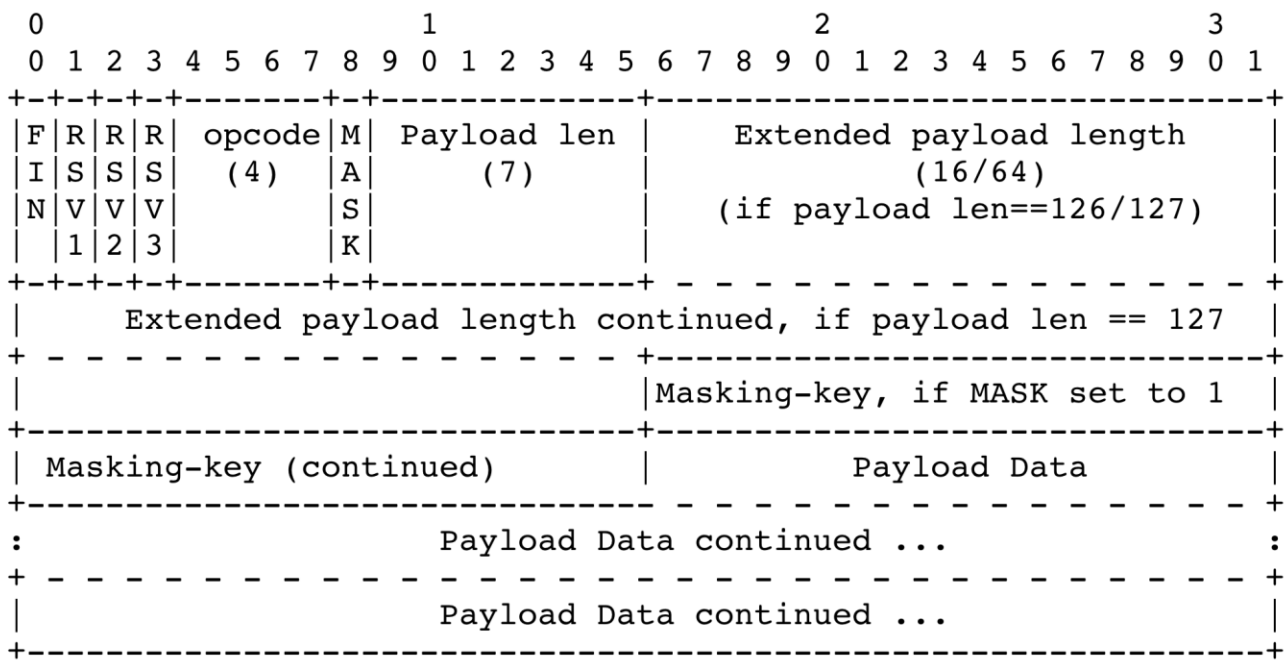

Рисунок 1.5 - Структура WebSocket фрейму за RFC 6455

Заголовок пакету складається з:

2. FIN (1 біт): маркер того, чи є поточний фрейм останнім. Виставляється і у випадку, коли повідомлення складається з одного фрейму.

3. RSV1, RSV2, RSV3 (кожний з параметрів складається з одного біта): дані поля повинні бути встановлені в значенні 0 за умови, що не було попередньої домовленості про розширеннях до значень даних параметрів. За умови, що у фреймі буде встановлено значення 1 без попередньої домовленості, отримувач повинен закрити з'єднання.

4. opcode (4 біти): кодування змісту фрейму. Використовуються такі значення, як:
   a. 0x00: в поточному фреймі міститься наступна частина повідомлення.
   b. 0x01: фрейм містить текстові дані (кодування – UTF-8)
   c. 0x02: фрейм містить бінарні дані
   d. 0x08: поточний фрейм завершує з'єднання
   e. 0x09: ping-фрейм
   f. 0x0a: pong-фрейм

5. mask (1 біт): вказує на те, що фрейм замаскований. За специфікацією, кожне повідомлення від клієнта до сервера повинно бути замасковано. В іншому випадку, специфікацією рекомендовано примусово розірвати з'єднання.

6. payload_len (7 біт): зберігає довжину корисного навантаження повідомлення. Значення 0-125 вказує на довжину корисного навантаження. Значення 126 вказує на те, що наступні 2 байти визначають розмір корисного навантаження. Значення 127 вказує про те, що наступні 8 байтів зберігають інформацію про розмір корисного навантаження повідомлення.

7. masking-key (32 біти): усі фрейми, що будуть відправлені із клієнта на сервер повинні бути замасковані з використанням 32-х бітного значення, що зберігається у фреймі.

8. payload: корисне навантаження повідомлення (довжина якого повинна дорівнювати значенню payload_len)

З використанням наведеної вище концепції та специфікації фреймів досягається гнучкість типізації корисного навантаження фреймів та його розмірів.

Для оптимізації пам'яті можна використовувати heartbeat-повідомлення (описані в специфікації RFC 6455). Основою heartbeat-повідомлень є концепція ping-pong повідомлень: сервер надсилає ping-повідомлення на клієнтську частину програмного продукту. Клієнт, у свою чергу, надсилає pong-відповідь на запит [8]. Таким чином, якщо клієнт відповів pong-повідомленням на ping-запит, з'єднання вважається активним.

З використанням фрагментації фреймів, серверна частина програмного продукту може використовувати буфер фреймів, за наповнення якого фрейми надсилаються на клієнтську частину програмного продукту, що можна використати для оптимізації використання мережевих ресурсів[9].

Найуразливішим місцем програмного продукту, (основним протоколом обміну даними якого є WebSocket), є серверна частина (рисунок 1.6). Клієнтська та серверна частина повинні відповідати за трансформацію даних та підтримку постійного зв'язку з сервером.

За технічної відмови сервера всі клієнти, що утримують зв'язок із сервером, втрачають зв'язок один з одним. Вирішенням даної проблеми є горизонтальне масштабування серверної частини програмного забезпечення, яке веде до збільшення як фінансових, так і ресурсних витрат на підтримку серверів.

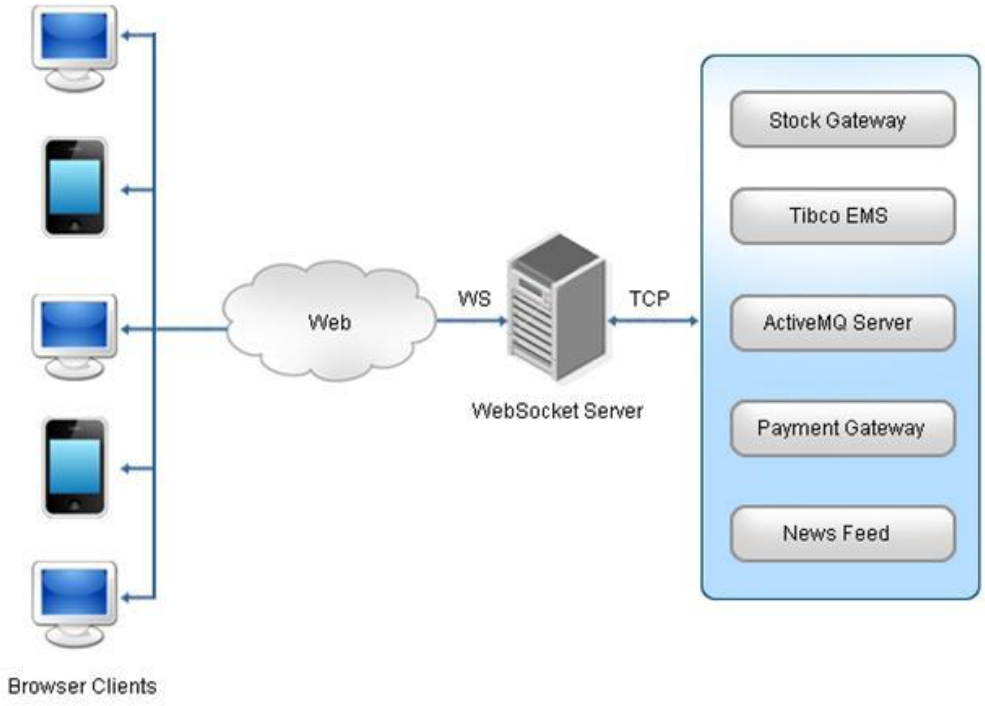

Рисунок 1.6 - Схема відеоконференцій без використання технології WebRTC

Перевагою WebSocket є відсутність обмеження на кількість як активних, так і пасивних з'єднань крім пам'яті самих серверів. Протокол зарекомендував себе з гарної позиції за багатьма параметрами, такими як безпека, універсальність та швидкість. [10].

Він реалізований у сучасних браузерах, що уможливлює його використання в проектах, основою яких є інтерактивна взаємодія з користувачем.

Мінусами використання технології є менеджмент потоків на серверній частині програмного продукту та постійна трансформації даних як на клієнтській, так і на серверній частині програмного продукту.

Серверна частина програмного продукту, що імплементує протокол WebSocket, є найуразливішим місцем програмного забезпечення.

Клієнтська частина програмного продукту повинна одночасно групувати повідомлення WebSocket та, з використанням паралелізації, трансформувати дані, перевіряти їх цілісність, актуальність та відповідати за вибір кодеків і трансформацію даний.

1.3. Протокол передачі даних WebRTC

Нова технологія WebRTC надає можливість досягнення компромісу вимог до швидкодії та витрат на підтримку серверної частини програмного забезпечення.

Технологія WebRTC регламентована у RFC-7478 [11] та RFC-5245 [12]. В них описані вимоги до розробників браузерів, що надають API використання імплементацій протоколів передачі WebRTC для розробників клієнтської частини програмного продукту.

Основною ідеєю WebRTC є імплементація налаштування зв'язку користувачів напряму за концепцією браузер-браузер. Таким чином, передача медіа-даних не потребує наявності серверної частини як такої. Головним зобов'язанням сервера є налаштування обміну конфігураційними даними між клієнтськими частинами програмного забезпечення (рисунок 1.7).

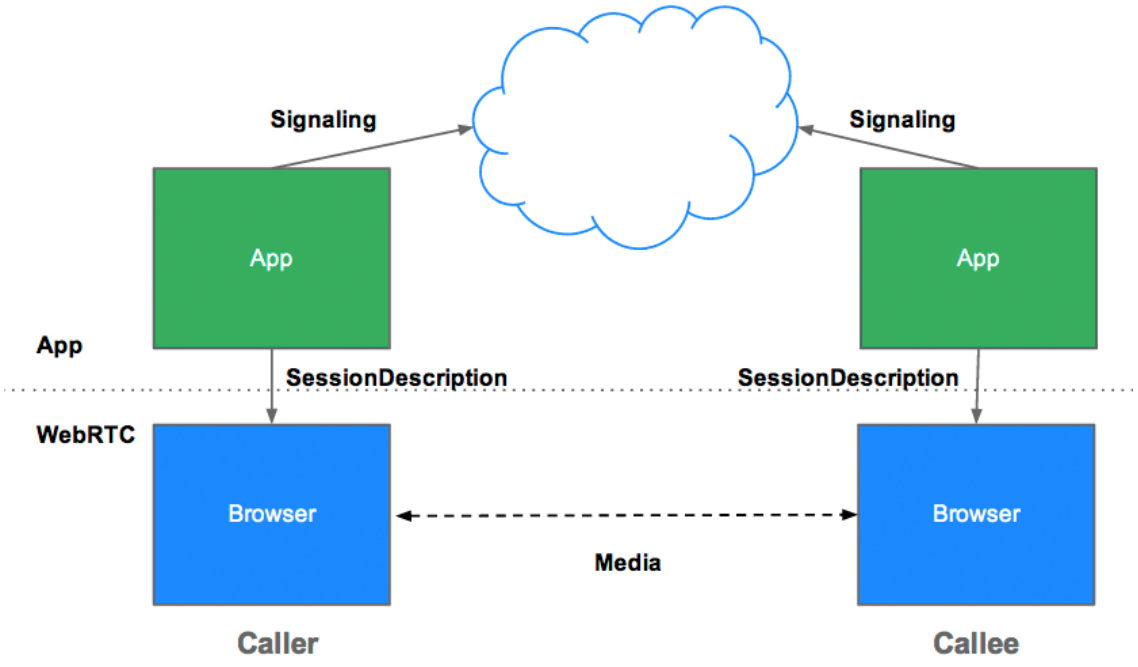

Рисунок 1.7 - Схема взаємодії клієнтів з використанням технології WebRTC

Перед початком обміну медіа-даними, клієнтські частини програмного продукту надсилають "запрошення" (offer) та "відповідь на запрошення" (answer on offer) за протоколом SDP [13]. Якщо відповідь на запит підключення отримана і має позитивний

характер, клієнти починають обмін кандидатами на підключення (ICECandidate) за протоколом ICE.

Серверна частина програмного продукту, за такої архітектури, виконує лише початкове налаштування зв’язку. Таким чином, навіть за повної відмови серверної частини програмного забезпечення, існуючі з’єднання не будуть знищені та будуть передавати інформацію без зміни якості та швидкодії (що не може бути реалізовано з використанням сімейств протоколів Polling та WebSocket).

Отже серверна частина програмного продукту вже не є слабким місцем системи та виконує роль лише налаштування з’єднання і не має керувати повним життєвим циклом каналів передачі медіа-даних.

Це досягається завдяки інтерфейсу RTCPeerConnection, що дозволяє налаштувати канал зв’язку між конкретними браузерами. За допомоги даного інтерфейсу існує можливість до налаштування каналу зв'язку між конкретними клієнтами системи, передаючи дані напряму між браузерами, розвантажуючи серверну частину програмного продукту, та надаючи можливість передачі між браузерами відео- та аудіо-потоків без додаткової розробки логіки трансформації та перекодування.

Сигнальний сервер, у свою чергу, повинен бути спроектований та реалізований з відповідністю з логікою обміну з клієнтською частиною програмного продукту.

Основною зоною відповідальності сигнального сервера є передача сигналів-налаштувань між клієнтськими частинами програмного продукту, що необхідні для налаштувань RTCPeerConnection. Формат передачі даних клієнт-сервер, за специфікацією, може бути обраний розробниками програмного забезпечення.

У момент обміну ICE-кандидатами, за специфікацією WebRTC, необхідно використовувати STUN-сервери. Без використання STUN-сервера, налагодження каналів передачі даних браузер-браузер буде можливим лише за умови, якщо обидва браузери фізично знаходяться в одній локальній мережі. (рисунок 1.8)

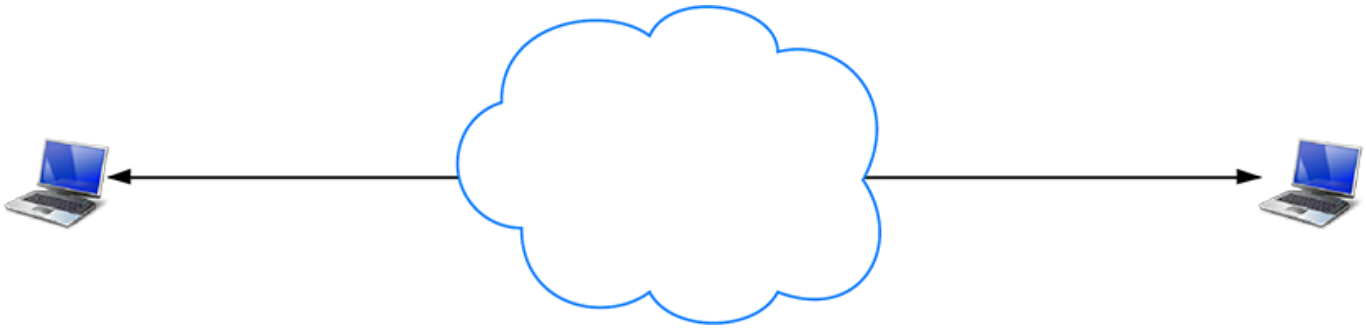

Рисунок 1.8 - Зв’язок браузерів у локальній мережі.

Протокол STUN — це мережний протокол, який дозволяє користувачу визначити свою зовнішню IP-адресу, спосіб трансляції адреси та порт у зовнішній мережі, пов'язаною із визначенням внутрішнього номеру порту.

Ця інформація використовується для встановлення з'єднання UDP (User Datagram Protocol — один із протоколів в стеку TCP/IP) між двома хостами у випадку, коли вони знаходяться за NAT-маршрутизатором [14].

Протокол визначено у специфікації RFC 3489. NAT (Network Address Translation) — це механізм зміни мережної адреси в заголовках IP датаграм, поки вони проходять через пристрій маршрутизації з метою роутингу одного адресного простору в інший. [16]

STUN-сервери розгорнуті у загальному доступі в мережі Інтернет та відповідають за розкриття публічної частини хостів клієнтів. Процес розкриття дозволяє веб-вузлу WebRTC одержати загальнодоступну адресу для себе, а потім передавати її іншій одноранговій системі через механізм сигналізації, щоб встановити прямий зв'язок (рисунок 1.9.) [17].

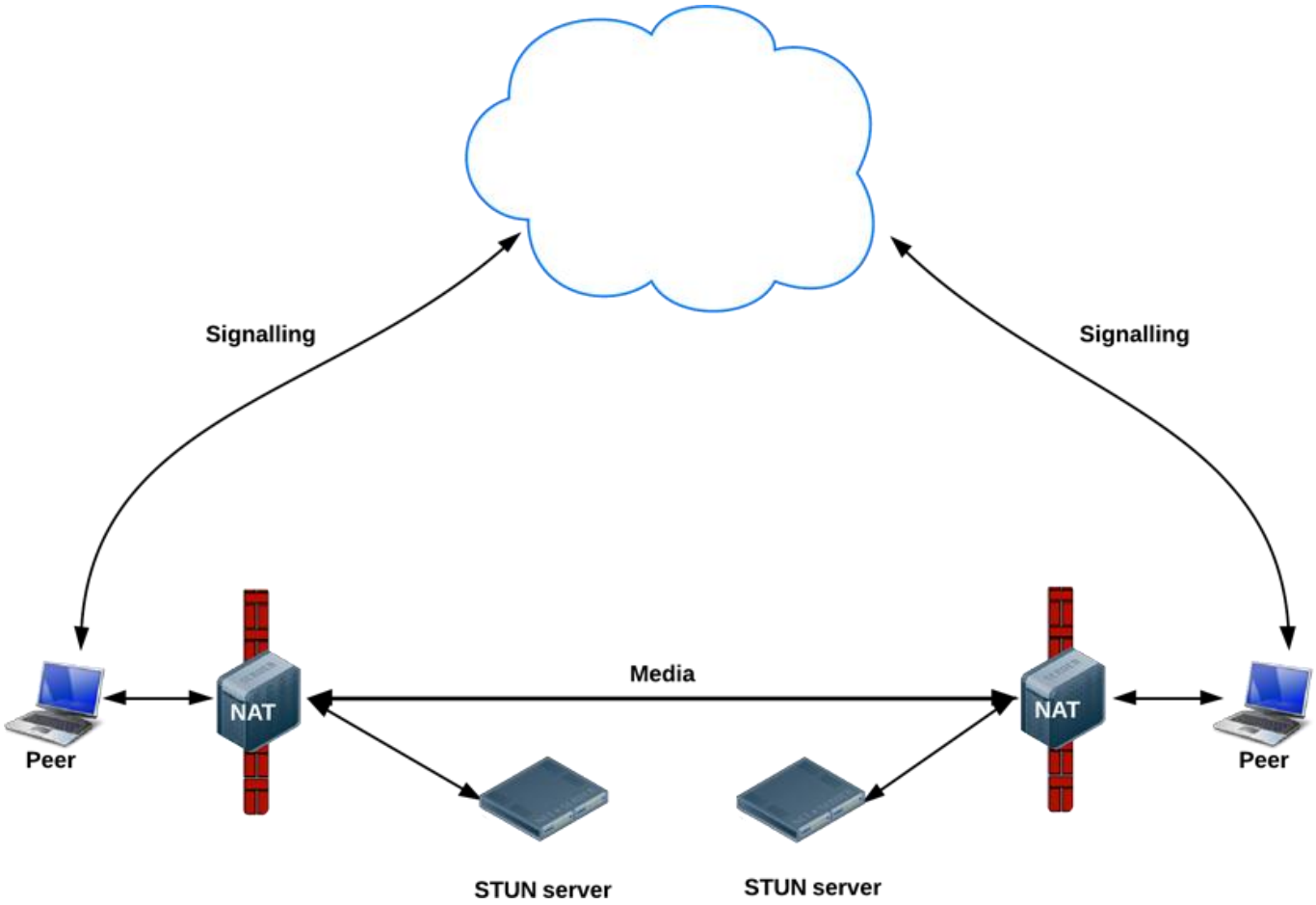

Рисунок 1.9 - Схема взаємодії браузерів з використанням STUN-серверів

За наявності STUN-сервера, система може налагодити зв’язок, але цього не достатньо для ретрансляції медіа-потоків між браузерами у випадку негативної відповіді від STUN-серверів.

Використання ретранслятора NAT TURN дозволяє вузлу за NAT або брандмауером отримувати вхідні дані через TCP або UDP з'єднання. Така можливість особливо актуальна для вузлів за симетричними NAT, або брандмауерів[17].

TURN сервери мають концептуальне завдання, що полягає у ретрансляції адрес потоків даних між точками доступу (рисунок 1.10).

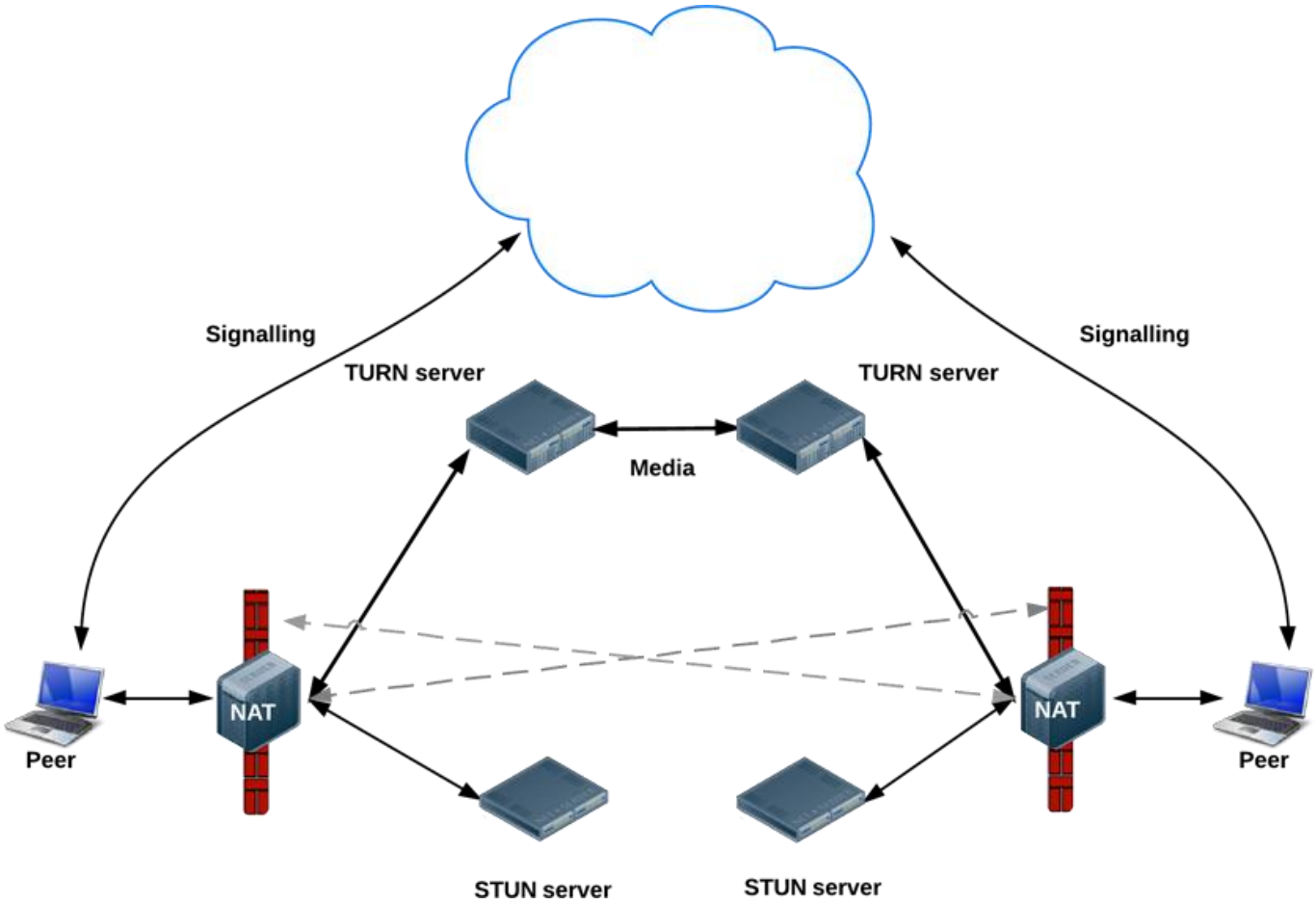

Рисунок 1.10 - Схема взаємодії браузерів з ретрансляцією потоків даних

Для забезпеченння обміну конфігураційними даними каналів передачі медіа-потоків між клієнтськими частинами програмного продукту доцільно використати протокол SSE, що є найефективнішим (за характеристиками швидкодії та ресурсоємності) у порівнянні з HttpStreaming та WebSocket.

Таким чином, відпадає необхідність у підтримці великих потужностях сервера та у грошових витратах для їх горизонтального масштабування, за рахунок використання сервера тільки для передачі конфігурацій налаштувань медіа-потоку.

1.4. Висновки до розділу 1

Порівнюючи такі протоколи передачі даних, як Polling, Long Polling, HTTP Streaming (SSE) та WebRTC, можемо дійти висновку, що відмова серверної частини у

концепціях Polling, Long Polling та HTTP Streaming (SSE) призводить до повної відсутності доступу до інформації всіх користувачів. Серверна частина стає вузьким місцем при забезпеченні передачі медіа-даних між багатьма користувачами.

З іншого боку, у випадку технології WebRTC, сервер використовується лише передачі конфігураційних даних каналів передачі медіа-потоків між клієнтськими частинами програмного продукту, але ніяк не для передачі даних поміж ними. З використанням протоколів WebRTC, серверна частина програмного продукту більше не повинна відповідати за отримання, трансформацію та передачу медіа-потоків, навантаження на серверну частину програмного продукту, за рахунок цього, значно зменшено.

Таким чином, зважаючи на переваги та недоліки існуючих технологій, обґрунтовано вибір технології WebRTC для реалізації передачі медіа-потоків в програмному продукті, що розроблюється.

Обгрунтовано вибір протоколу Server-Sent Events для забезпечення передачі конфігураційних даних, необхідних у процесі з'єднання за протоколом WebRTC.

2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ ЗАБЕЗПЕЧЕННЯ ВІДЕОКОНФЕРЕНЦІЙ

Архітектура програмного забезпечення є базисним компонентом на етапі розробки програмного продукту. За ANSI/IEEE 1471-2000, “Архітектура – це базова організація системи, втілена в її компонентах, відношеннях між ними та оточенням та принципи, що визначають проектування і розвиток системи”. Під системою мається на увазі набір компонентів, об’єднаних для виконання визначеної функції або їх набору. Архітектура програмного забезпечення включає в себе організацію взаємозв’язків між програмними компонентами та зовнішнім середовищем із елементами програмного продукту.[18]

Згідно з проведеним аналізом технологій у розділі 1, з використанням протоколів WebRTC, серверна частина програмного продукту більше не повинна відповідати за отримання, трансформацію та передачу медіа-потоків. Також, підхід розробки програмного продукту з використанням технології WebRTC є менш ресурсномістким з точки зору використання порівняно з іншими технологіями, розглянутими у розділі 1.

Архітектура клієнт-сервер є одним із архітектурних шаблонів, що поступово перетворюється в стандарт де-факто у промисловості розробки програмного забезпечення.

З огляду на те, що основною функціональністю програмного продукту є забезпечення відео-зв’язку користувачів з використанням технології WebRTC, основним компонентом системи є клієнтська частина програмного продукту, оскільки саме функціональність клієнтської частини програмного забезпечення відповідає за налаштування передачу даних між браузерами.

Далі проводиться аналіз методів проектування систем та архітектурних шаблонів блоки побудови архітектури програмного забезпечення та розроблено

архітектури програмного продукту, що розроблюється. Виконується декомпозиція поведінки системи на блоки та проектування архітектури системи.

2.1. Шаблони проектування

На сьогоднішній день, всесвітня мережа Інтернет зазнала стрімкого розвитку та поширення, що зумовило необхідність стандартизованого та одночасно гнучкого інструменту взаємодії з користувачем.

Шаблони проектування описують типові способи вирішення поширених проблем при проектуванні програмного забезпечення. Патерни проектування були вперше описані Крістофером Олександром в книзі Мова шаблонів: міста, будівлі, конструкції (Christopher Alexander, A Pattern Language: Towns, Buildings, Construction, Oxford University Press). У ній була введена концепція, названа шаблоном, - абстрактне рішення часто виникаючих проблем проектування, що привернула в середині-кінці 1980-х років увагу дослідників в інших областях, зокрема, архітекторів та розробників програмного забезпечення[19].

Дослідження шаблонів проектування в програмному забезпеченні еволюціонували після виходу, можливо, найзначнішої книги з об'єктно-орієнтованого проектування: Шаблони проектування: Елементи повторно використовуваного об'єктно-орієнтованого програмного забезпечення, написаної Еріком Гамма, Річардом Хелмом, Ральфом Джонсоном і Джоном Вліссідес (Erich Gamma, Richard Helm , Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley). [20]

Найбільшою частиною Design Patterns є каталог, що описує 23 шаблони проектування. Інші, більш пізні версії каталогів шаблонів проектування, лише розширюють цей набір і, найважливіше, розширюють його сферу дії більш спеціалізованими типами задач. Каталог повторно використовуваних шаблонів проектування, ілюстрований з використанням UML.

Саме з використанням існуючих шаблонів проектування було побудовано архітектуру програмного забезпечення.

Архітектура клієнт-сервер складається з трьох основних компонентів: набір сервісів, які зберігають та надають інформацію, що до них звертаються, набір клієнтів, що використовують набір сервісів, наданих серверам та, відповідно, мережа, що має за мету створення каналу взаємодії між клієнтами та серверами[21].

Сервери і клієнти є незалежними один від одного і функціонують паралельно. Клієнти та сервери пов’язані один з одним за принципом запит-відповідь, що передбачає гнучкість та незалежність модулів глобальних світових мережевих веб-систем.

Під час аналізу проектування майбутнього серверного програмного продукту було обрано за основу архітектурний концепт MVC (Model-View-Controller), що дозволяє розмежувати клієнтську та серверну частини програмного продукту. Він був обраний з розрахунку на широку сферу впливу в області сучасної розробки клієнт-серверних програмних продуктів.

Основна ідея принципу MVC полягає в тому, що будь-який програмний продукт у першому наближенні можна розподілити на два модулі: бізнес-логіку та представлення, що має на меті створити зв'язок функціональності та користувача [22].

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає розподіл прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка застосунку [23].

Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверних застосунків, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Управління даними здійснюється сервером даних.

2.2. Проектування архітектури клієнтської частини програмного продукту з використанням технології WebRTC

Обрана технологія WebRTC передбачає налагодження зв'язку між клієнтами програмного продукту, використовуючи можливості браузеру. Таким чином, варто розглянути архітектуру побудови клієнтської частини програмного продукту, оскільки саме вона накладає вимоги до серверної частини програмного забезпечення.

Для забезпечення постійного зв'язку, клієнтська частина програмного продукту повинна постійно зберігати у пам'яті об'єкти RTCPeerConnection, що є інтерфейсом каналу передачі медіа даних між браузерами. Враховуючи це обмеження та необхідність збереження інтерактивності клієнтської частини, необхідно здійснити побудову архітектури клієнтської частини, що не допускала б перезавантаження сторінки браузеру.

Вирішенням даної проблеми є вибір технології розробки клієнтської частини за принципом SPA: Single Page Application, що використовує єдину HTML сторінку в якості оболонки для підрядних частин DOM-моделі.

Таким чином, під час зміни бізнес-моделей та моделей відображення, змінюється лише підрядна модель сторінки.[24]

Розробники клієнтської частини програмного забезпечення, що не є функціонально статичними, використовують мову програмування JavaScript завдяки повній підтримці мови браузерами [25].

Стандартом, що специфікує мову програмування JavaScript та її версіонування є ECMAScript. [26]

Першою версією стандарту є ES1. Зараз версія стандарту ES5, що була випущена у 2009 році, має повну підтримку браузерів[25].

В даний час існують такі версії стандарту[27]:

1. ECMAScript 1 (1997) – перший випуск стандарту;
2. ECMAScript 2 (1998) – невеликі зміну випуску;
3. ECMAScript 3 (1999) – додані Regular Expressions та підтримка try/catch;

4. ECMAScript 4 – версія не випущена;
5. ECMAScript 5 (2009) – додано “strict mode”, підтримка JSON;
6. ECMAScript 6 (2016) – додані ключові слова let, const;
7. ECMAScript 7 – доданий exponential operator;
8. ECMAScript 8 (2017) – додані async-функції та shared memory оброрбники;
9. ECMAScript 9 (2018) – додані Promise.finally()

Одним із найрозповсюдженіших фреймворків побудови клієнтських частин програмного забезпечення за принципом SPA, що може бути скомпільовано у мову програмування JavaScript з підтримкою специфікації ES5 та синтаксичною схожістю на основні концепції зі специфікаціями ES6-ES8 є фреймворк Angular (версії 5+).

Фреймворк Angular версій 5+ використовує мову TypeScript, що, в свою чергу, є надбудовою над JavaScript специфікацій ES6-ES8. Він дозволяє створювати динамічні інтерактивні WEB-сайти з підтримкою для більшості мобільних пристроїв та браузерів.

Основними будівельними блоками фреймворку є модулі Angular – NgModules, які надають контекст компіляції для компонентів. Модулі NgModules аналізують та агергують відповідний код у функціональні набори, з яких складається застосунок Angular. Клієнтська частина програмного забезпечення, що побудована на базі фреймворку, має принаймні одне об’явлення NgModule, який має налаштування розгортування додатку на стороні клієнта. Головний модуль агрегує в собі інші функціональні модулі фреймворку.

Компоненти визначають компоненти відображення, визначені мовою розмітки HTML та CSS, для відображення та зміни компонентів відображення відповідно до написаної бізнес-логіки та зв’язаних моделей даних.

Компоненти використовують Angular-сервіси (Service), що надають особливі функції, що не пов’язані із представленнями. Сервіси фреймворку можуть надавати можливість до агрегації сервісів у компонентах з використанням концепції Dependency Injection, що робить програмний код модульним, ефективним та багаторазовим [28].

Компоненти та сервіси визначені класами з використанням декораторів та визначення метаданих.

Основні блоки фреймворку Angular надають гнучкий інструментарій для побудови клієнтської частини програмного продукту. Клієнтська частина програмного забезпечення побудованого з використанням фреймворку повністю задовольняє вимогам Single Page Application та надає гнучкий інструментарій для побудови клієнтської частини програмного продукту.

Після вибору технології побудови фундаменту клієнтської частини програмного продукту, необхідно визначити вимоги для використання технології WebRTC у якості компоненту клієнта продукту, що розроблюється.

Першим блоком для побудови відеоконференцій є побудова двостороннього зв'язку клієнт-клієнт з використанням технології WebRTC. (рисунок 2.1)

Для прикладку візьмемо двох користувачів системи: Alice та Bob, що намагаються налагодити відео-зв'язок один з одним.

Клієнт Alice авторизується у системі, використовуючи свій логін-пароль та оформлює підписку на події сервера, за технологією Server-Sent Events. Клієнт Bob також авторизується у системі та оформлює підписку на події сервера (авторизація може бути виконана у різний період часу).

Клієнт Alice вирішує надіслати пропозицію на розмову з клієнтом Bob та ініціює запит на розмову із іншим клієнтом, використовуючи веб-інтерфейс.

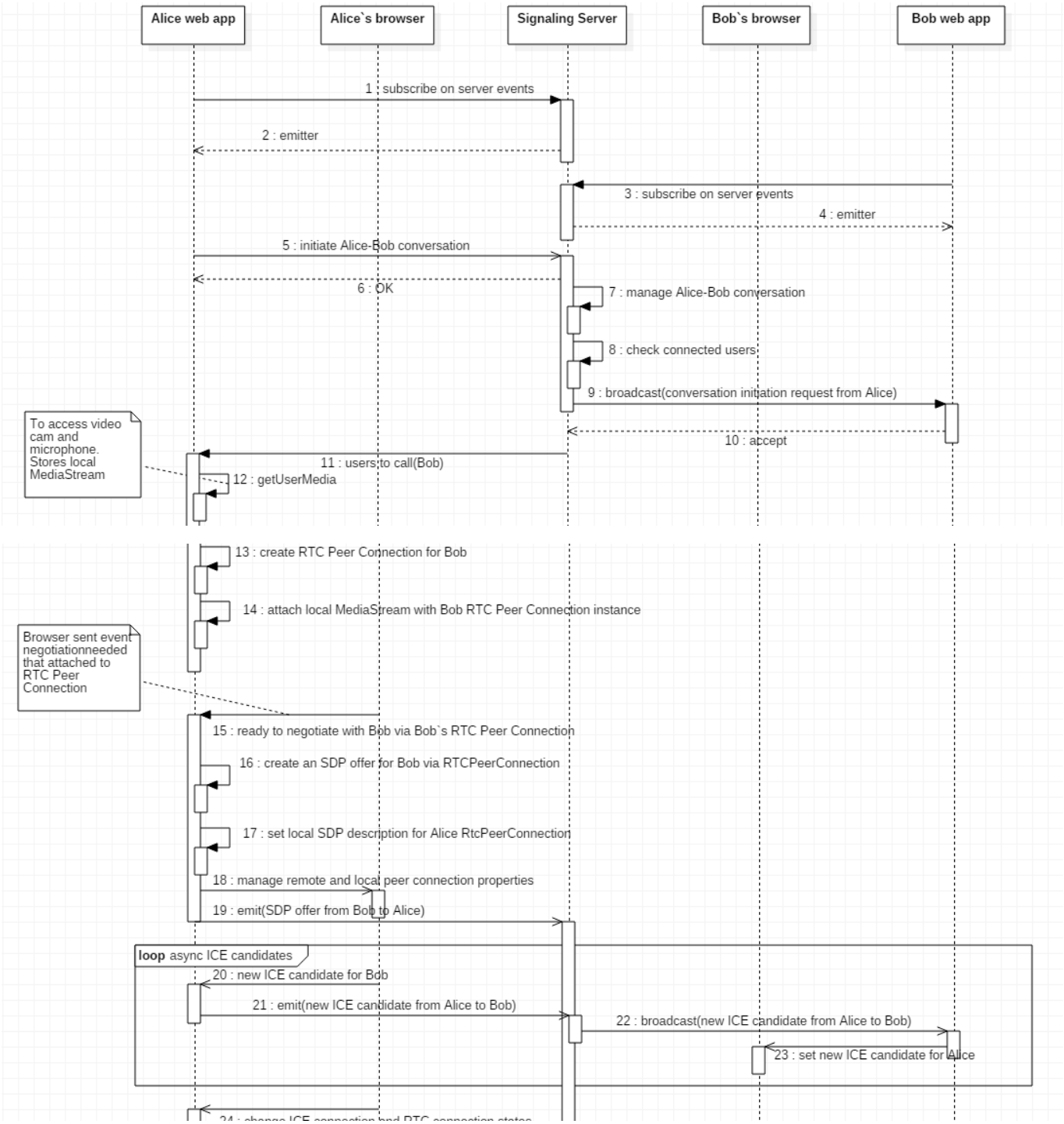

Рисунок 2.1 - Діаграма послідовностей налагодження каналу зв’язку за технологією WebRTC. Частина 1.

Даний запит надходить на сигнальний сервер (частина серверної частини програмного продукту), що аналізує отриманий запит на предмет наявності отримувача запиту у системі, та, у разі виявлення факту, що клієнт Bob онлайн,

надсилає (з використанням сигнального сервера) до Bob запит на розмову з Alice із метаінформацією розмови.

Клієнт Bob, у свою чергу, підтверджує запит на розмову з Alice та надсилає нове повідомлення-підтвердження на сигнальний сервер.

На даному етапі можна стверджувати, що обидва клієнти (Bob та Alice) у стані "онлайн" у системі, авторизовані, підтвердили розмову один з одним та мають можливість до налагодження зв'язку один з одним. Після цього етапу розпочинається налагодження зв'язку між браузером Alice та браузером Bob з використанням веб-частини програмного забезпечення.

Метаінформація щодо налаштувань браузерів обох та метаінформація каналу передачі відео та аудіо потоків клієнтів повинна бути інкапсульованою в об'єктах клієнтської частини програмного продукту і повинна бути ініційована лише після авторизації в системі та отримання дозволу обох клієнтів на поширення інформації (рисунок 2.2).

Саме архітектурний принцип SPA забезпечує зберігання налаштувань каналу передачі браузер-браузер без необхідності до повторного процесу налаштування каналу передачі даних за взаємодії з програмним продуктом без необхідності до перезавантаження веб-сторінки.

Наступними етапами до налаштування каналу передачі даних між браузерами є надсилання WebRTC запиту, відповіді, та надсилання налаштувань ICE кандидатів, в яких агрегована інформація про налаштування браузерів та налаштування доступу до них.

З цих етапів можемо дійти до висновку щодо необхідності побудови окремих веб-компонентів клієнтської частини та налаштування асинхронної підписки на події сервера.

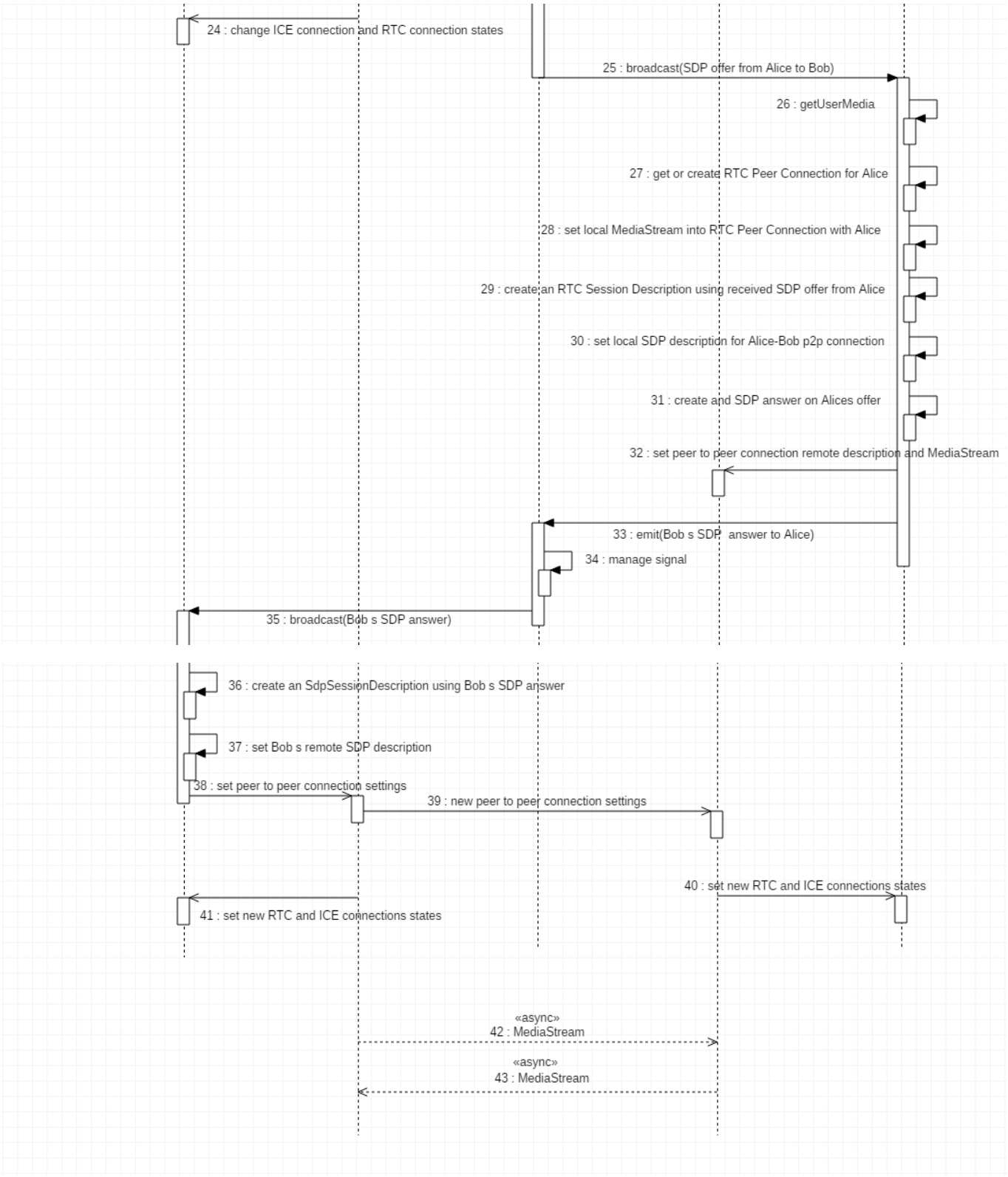

Рисунок 2.2 - Діаграма послідовностей налагодження каналу зв’язку за технологією WebRTC. Частина 2.

Для забезпечення відеоконференції необхідно реалізувати вище описаний сценарій для всіх користувачів системи, що беруть участь у відео конференції. Таким чином, в розрізі однієї відеоконференції, для кожного учасника розмови буде налаштовано канал передачі даних з кожним браузером інших учасників розмови.

Дана інформація встановлює вимоги та обмеження до архітектури клієнтської частини програмного забезпечення. На рисунку 2.3 зображено ієрархію функціональних блоків, що відповідають за налаштування з'єднань браузер-браузер. Кожний об'єкт WebRTC інкапсулює методи створення каналу браузер-браузер, його життєвий цикл та зберігає медіа-потік для можливості відображення відео-блоку на прошарку відображення.

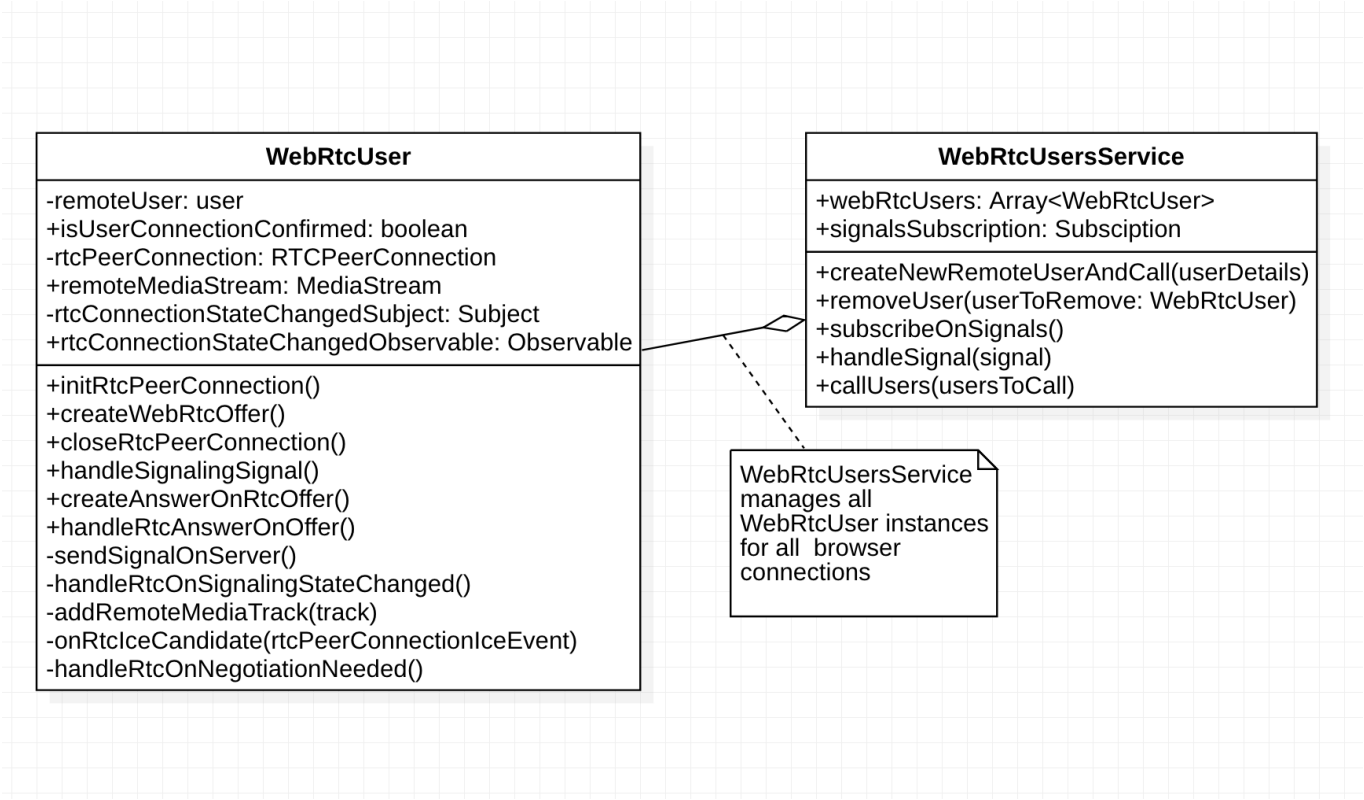

Рисунок 2.3 - Взаємодія класів, що відповідають за налаштування WebRTC з'єднань

Сервіс WebRtcUsersService є сервісом в термінології фреймворку Angular, об'єкт якого ін'єктовано у глобальній області видимості веб-застосунку клієнта. Таким чином, уся логіка налаштування каналів зв'язку браузер-браузер інкапсульовано у цих двох Angular-класах.

Основною функціональністю сервісу є управління усіма користувачами, з'єднаними із поточним користувачем. На рисунку 2.4 схематично відображено архітектуру клієнтської частини програмного продукту у вигляді діаграми пакетів.

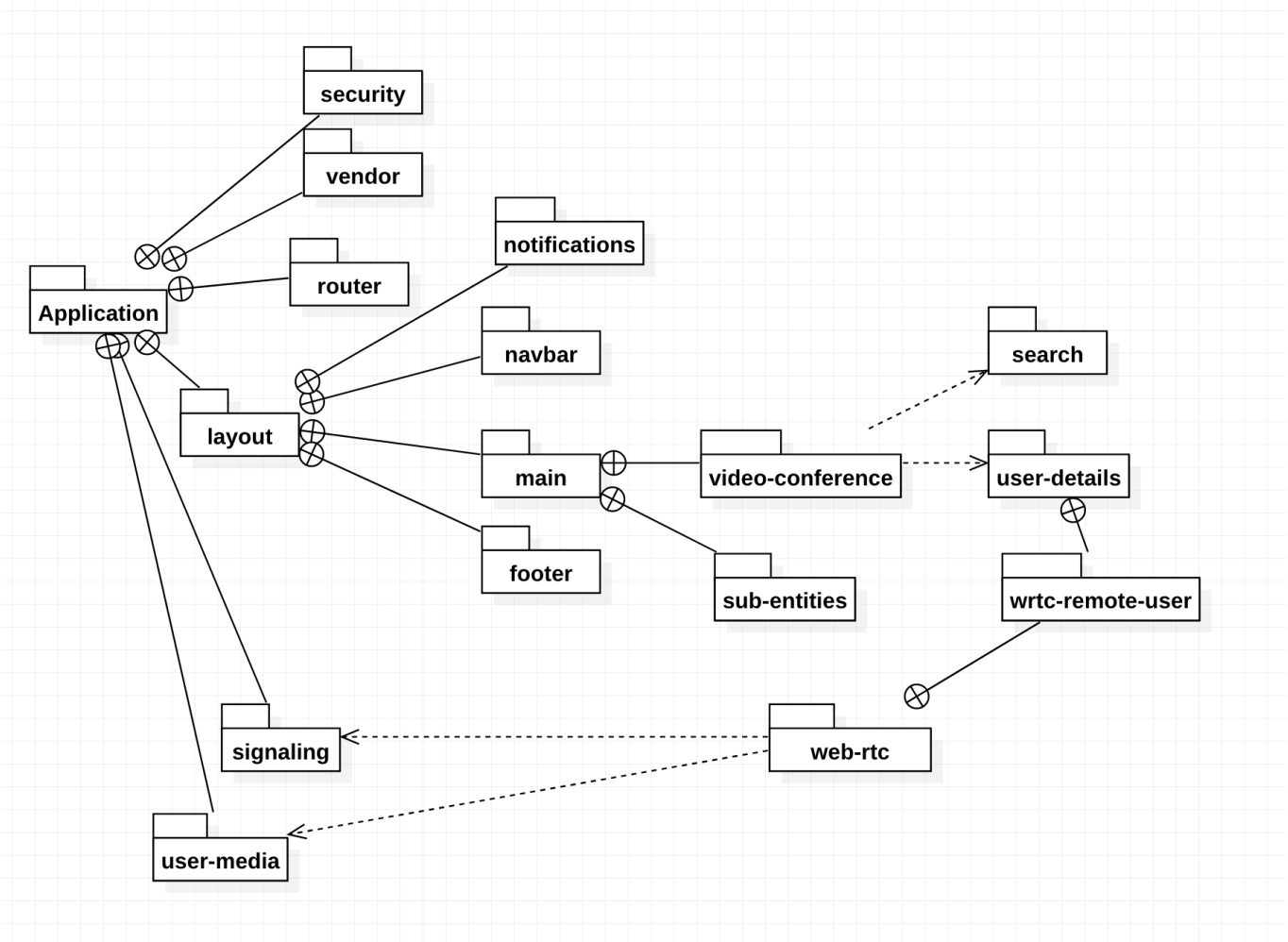

Рисунок 2.4 - Діаграма пакетів клієнтської частини програмного продукту.

Кожний пакет містить відповідний Module, Service, Component, CSS-файл та HTML файл (за наявності моделі відображення) та сервіси роутингу сторінки. Структуризація за пакетами надає можливість створення модульної системи, що, в разі необхідності, може бути змінена.

Таким чином, було проведено декомпозицію клієнтської частини програмного продукту та визначено основні пакети та класи, що повинні відповідати за налагодження каналу зв'язку для забезпечення відеоконференцій.

2.3. Проектування архітектури серверної частини програмного продукту з використанням технології WebRTC

Сучасний технологічний розвиток програмного забезпечення значною мірою вплинув на розвиток методології та основних принципів пошуку інформації в мережі Internet. Це каталізувало розвиток середовищ та технологій WEB-програмування.

На думку М. Фаулера: “Вибір мови програмування обумовлений тільки ймовірними уподобаннями читачів. Java в цьому сенсі виглядає найбільш привабливо” [29].

Цю ж думку з приводу використання Java як мови програмування підтримує і Н. Вязовик, яка зауважує, що “Java широковідома як сучасна об’єктно-орієнтована мова, легка у вивченні та така, що дозволяє створювати програми, які можуть виконуватись на будь якій платформі без доробок (кросплатформність). Також відомо, що Java орієнтована на Інтернет” [30].

Саме з розрахунку переваг мови програмування Java, її було обрано в роботі.

Під час аналізу архітектури майбутнього серверного програмного компоненту було обрано за основу архітектурний концепт MVC (Model-View-Controller), що дозволяє розмежувати клієнтську та серверну частини програмного продукту. Він був обраний з розрахунку на широку сферу впливу в області сучасної розробки клієнт-серверних програмних продуктів.

Основними зобов’язаннями серверної частини програмного продукту є:

1. Функція сигнального сервера;
2. Авторизація/автенттифікація;
3. Менеджмент активних відеоконференцій;
4. Логіка обміну повідомленнями між користувачами системи.

Основними обов’язками сигнального сервера, що є частиною серверної частини програмного продукту, є прийом/відправка подій користувачів системи. Кожний клієнт надсилає HTTP запит події на сервер, який, у свою чергу, аналізує прийнятий

від користувача сигнал, обробляє його, та, у разі необхідності, надсилає відповідну подію кожному учаснику процесу створення відеоконференції.

Технологія WebRTC передбачає побудову каналу зв'язку браузер-браузер. Для передачі конфігураційних даних з'єднання використовується сигнальний сервер. У роботі рекомендується використання технології Server-Sent Events для досягнення компромісу між швидкодією системи та навантаженням на сервер та мережу.

За обробку сигналів повинні відповідати сервіс управління потоками SSE та сервіс керування станами існуючих відеоконференцій, та станами з'єднань пар клієнт-клієнт.

Після визначення концептів спілкування клієнт-сервер та сервер-клієнт, необхідно зробити декомпозицію серверної частини програмного продукту.

Для використання функціональності системи, клієнт повинен бути зареєстрованим. Для цього необхідно пройти етап реєстрації, заповнення інформації (ім'я, прізвище, дані поштової скриньки та логін). Після реєстрації на поштову скриньку користувача буде надіслано лист із гіперпосиланням на активацію акаунту.

Таким чином, було імплементовано реєстрацію у два фактори: у системі та через поштову скриньку.

Після підтвердження реєстрації користувач має можливість до автентифікації у системі. Після успішної автентифікації у системі користувач отримає доступ до відповідних модулів функціональності системи, залежно від ролі користувача. Це дозволяє принципам безпеки даних користувачів, історію повідомлень та відео дзвінків.

Автентифікований користувач має можливість створення нової "дружби" з іншим користувачем системи та участі у відео конференціях з іншими користувачами системи після надсилання відповідного запиту.

Усі дані користувачів, пар спілкування, повідомлення та кімнати відеоконференцій зберігаються у базі даних (рисонок 2.5).

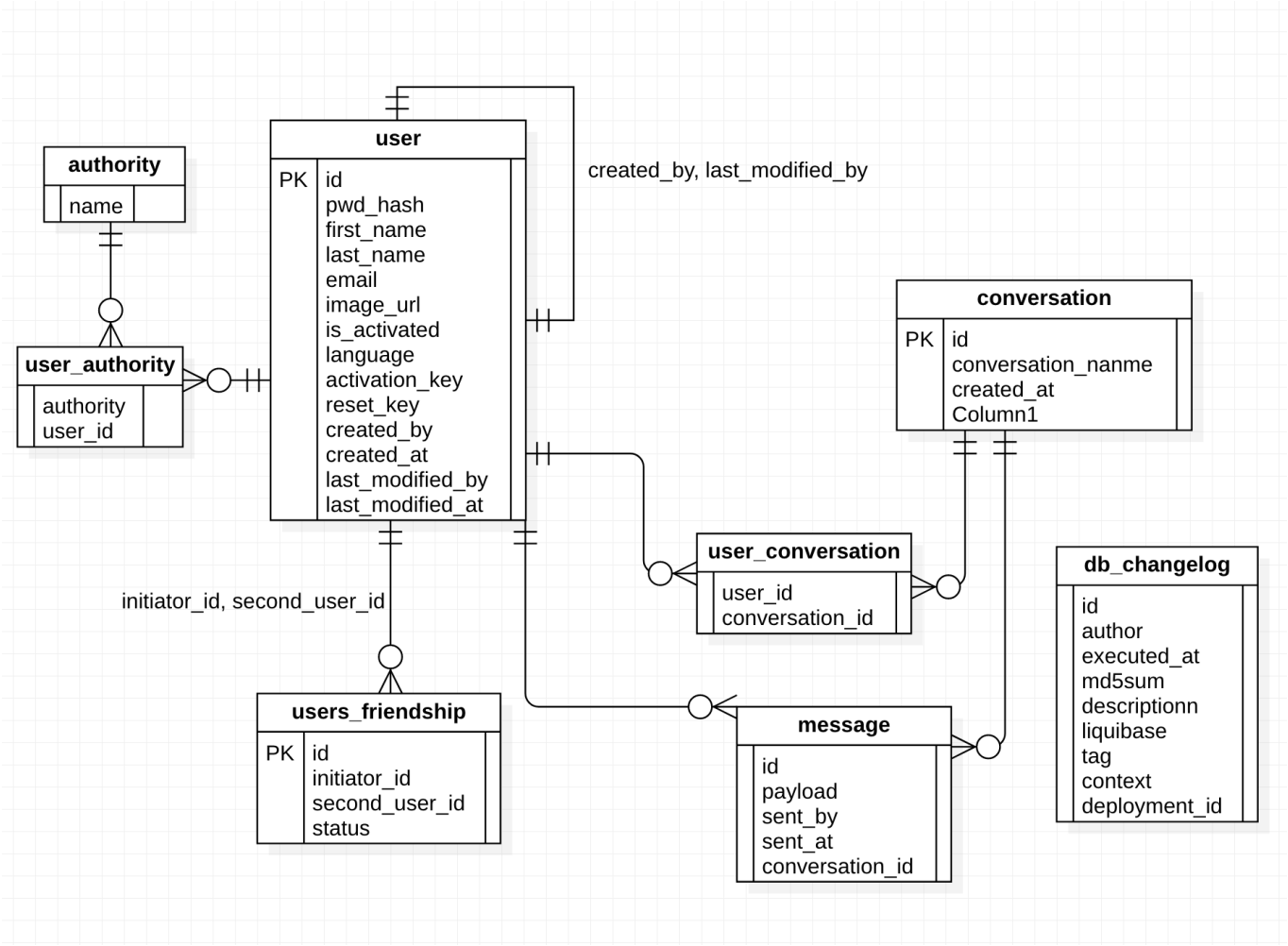

Рисунок 2.5 - Entity Relationship діаграма сутностей БД

Після підтвердження в участі у «розмові» (conversation), користувач має можливість до надсилання запиту на створення відеоконференції з усіма учасниками «розмови». У цей час, сервер має обробити запит, перевірити, чи обраний conversation вже активний і створити та зберегти в оперативній пам'яті.

Після чого, система повинна асинхронно перевірити активні з'єднання із кожною парою учасник-учасник конференцій. Якщо пари не знайдено, система створює в оперативній пам'яті пару ініціатор-учасник та переходить до наступного етапу. Якщо учасник онлайн – система, з використанням сервісу сигналізації, надсилає запит на підтвердження відеоконференції.

У випадку, коли учасник відеоконференції не знаходиться онлайн, система створює чергу повідомлень (якщо вона ще не існує) для користувача та зберігає повідомлення. В момент автентифікації клієнта у системі клієнтська частина

програмного продукту автоматично створює підписку на події сервера. Після успішної підписки на події, сервер аналізує черги повідомлень для користувача та надсилає відповідні сигнали-повідомлення.

За умови, що інший користувач приймає запит на відео конференцію, клієнтська частина програмного продукту в автоматичному режимі повинна надіслати відповідний сигнал на сигнальний сервер.

Сигнальний сервер, у свою чергу, аналізує користувачів, з якими необхідно налаштувати канал з'єднання і розпочинає обмін конфігураційними сигналами за протоколом WebRTC кожному з користувачів.

Таким чином, з використанням сигнального сервера та бізнес-логіки серверної частини програмного продукту, розпочинається налаштування каналів браузер-браузер усіх активних учасників відео конференції.

Після налаштування каналів зв'язку, сервер більше не бере участі у передачі медіа-потоків між користувачами відео конференції, тому, за технічної відмови сервера, налаштовані з'єднання не будуть перервані та не змінять показники швидкодії і якості потоків відео та аудіо.

Також, в роботі пропонується використання реляційної бази даних Postgres та фреймворку скриптів та версіонування схем баз даних Liquibase.

Liquibase - це бібліотека, що не залежить від бази даних з відкритим кодом, для відстеження, керування та застосування змін схеми баз даних. Бібліотека створена для легшого моніотрингу та змін у схемах баз даних [31].

Отже, було розроблено архітектуру основних блоків серверної частини програмного забезпечення.

2.4. Висновки до розділу 2

В цьому розділі було обгрунтовано вибір архітектурних концептів програмного комплексу, обрано шаблони проектування GOF та шаблон архітектури Model-View-Controller як базисний для програмного продукту, що розроблюється.

Було розглянуто особливості побудови клієнтської частини програмного продукту з використанням технології WebRTC, побудовано схему взаємодії клієнтських частин програмного продукту, інкапсульовано логіку взаємодії за WebRTC та визначено напрям та технології для подальшої програмної реалізації.

Визначено обов'язки серверної частини програмного продукту, визначені основні напрями імплементації, вимоги до серверної частини програмного продукту та до схеми бази даних.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ ЗАБЕЗПЕЧЕННЯ ВІДЕОКОНФЕРЕНЦІЙ

3.1. Програмна реалізація серверної частини програмного забезпечення

Після визначення мови програмування, вибору шаблонів програмування та глобального архітектурного концепту виникла необхідність проведення аналітики та вибору каркасу технологій програмування.

Після визначення базисного фреймворку, необхідно розробити прошарки серверної частини програмного продукту. Архітектурно, серверна частина повинна мати такі прошарки: [32]

- Прошарок контролерів – відповідає за виведення інтерфейсів (provided interface у термінології М.Фаулера). Пропонується використання Spring Core + Spring MVC;

- Прошарок бізнес-логіки – в ньому агреговано класи-сервіси бізнес-логіки, менеджмент транзакцій та сесій (Spring Core). Саме концепт MVC об'єднує клієнтську та серверну частини програмного забезпечення;

- Прошарок роботи з базисними даних (persistence layer) – прошарок, що відповідає за обмін даними між серверною частиною програмного продукту і засобами збереження даних (базами даних) (Spring Data) [33].

Необхідною частиною технології WebRTC є сигнальний сервер, що відповідає за обмін конфігураційними даними між клієнтськими частинами програмного продукту. Основною властивістю сигнального сервера повинна бути швидкість передачі сигналів. Таким чином, доцільно мати постійний канал передачі даних типу сервер-клієнт, що мав би на меті досягнення швидкості прийому та надсилання повідомлень-сигналів.

Використання протоколу передачі даних WebSocket передбачає побудову єдиного каналу зв'язку між компонентами системи, оскільки це є основною вимогою протоколу. Основними характеристиками за протоколом WebSocket є використання єдиного каналу зв'язку, що обробляє повідомлення направлення сервер-клієнт та клієнт-сервер[34].

У випадку побудови сигнального сервера, імплементація технології WebSocket передбачає побудову додаткового навантаження на мережу та ресурсні затрати сервера, що є надлишковим за рахунок кількості сигналів, що надходять із клієнта на сервер. Ще одним мінусом використання технології WebSocket у якості каналу сигналізації є складність налаштування імплементації протоколу як на серверній, так і на клієнтській частинах програмного забезпечення [37].

В роботі пропонується використання технології передачі Server-Sent Events. За цією концепцією, клієнтська частина програмного продукту оформляє підписку на події сервера після успішної автентифікації у системі.

Таким чином, з використанням технології Server-Sent Events, клієнтська частина програмного продукту оформлює підписку на повідомлення сервера, що є менш затратним за показниками навантаження на мережу та має властивість миттєвого прийому подій сервера напряму сервер-клієнт [36].

Події типу клієнт-сервер надсилаються на сигнальний сервер у вигляді REST HTTP запитів.

Після визначення основної технології, необхідно обрати концепцію реалізації технології Server-Sent Events. Основними можливими концепціями реалізації були:

1. Написання власного обробника HttpServletResponse і написання відповідного висхідного коду для обробки потоків;
2. Використання фреймворку Project Reactor та вихідного результату у вигляді Flux<ServerSentEvent<?>>.
3. Використання SseEmitter фреймворку Spring.

Використання Project Reactor, що є обгорткою для SSE вимагає перевизначення логіки FluxSink для можливості утримання каналу зв'язку без необхідності до повторної підписки зі сторони subscribe-рів. В обгортці кожного потоку з'являється

необхідність у створенні EventListener-ів в області видимості серверної частини програмного продукту.

Проблемами даного підходу є постійний менеджмент EventListener-ів, час життя яких не обмежується відкритим каналом передачі даних. Таким чином, в оперативній пам'яті сервера будуть існувати об'єкти EventListener-ів, які будуть виконувати бізнес-процеси для потоку даних, хоча сам потік даних вже вважається закритим як на стороні сервера, так і на стороні клієнта програмного забезпечення.

Використання ж функціональності фреймворку Spring (класу SseEmitter), приведе до збільшення ефективності витрат ресурсів оперативної пам'яті та дає можливість до написання більш зрозумілого і чистого програмного коду. Таким чином, в процесі підписки зі сторони клієнтської частини програмного продукту, RestController повертає об'єкт SseEmitter для конкретного потоку.

Менеджмент та життєвий цикл об'єктів, за такого підходу, стає обов'язком сервісу у прошарку бізнес-логіки серверної частини програмного забезпечення.

Таким чином, після отримання можливості до налагодження комунікації клієнтів, було отримано можливість налагодження каналу передачі даних між двома клієнтами системи.

Наступним етапом є програмна реалізація алгоритму створення та управління відеоконференціями між багатьма користувачами системи. Необхідно зауважити, що в понятті системи, що розроблюється, відео-розмова 1-1 також вважається відео конференцією з двома користувачами системи.

Так, як побудова каналів передачі даних за технологією WebRTC передбачає розподіл клієнтів на ініціатора та користувача, що приймає канал, необхідно розробити бізнес-логіку сервера для розподілу учасників відеоконференції на ініціаторів та споживачів та передбачити розподіл цих пар за технічною та логічною складовими.

За складовою бізнес логіки, один користувач приймає (або відхиляє) запрошення на створення відеоконференції один раз (для однієї відеоконференції). В іншому випадку, за умови великої кількості учасників відеоконференції, клієнт буде зобов'язаний приймати або відхиляти запрошення на створення відеоконференції для

кожного учасника, що суперечить прийнятим правилам побудови інтуїтивно зрозумілого інтерфейсу.

З іншого боку, технологія WebRTC передбачає дотримання чіткого розподілу клієнтів на ініціатора та того, що приймає пропозицію на створення каналу та відповідає власними конфігураційними даними.

Також, однією із важливих проблем, що може виникнути, є ініціація відеоконференції, інші учасники якої знаходяться офлайн. Таким чином, якщо не вирішити наявну колізію логіки та можливі ситуації поведінки користувачів системи, програмний продукт може реагувати непрогнозованим чином.

Враховуючи вищезазначене, пропонується вирішення даних проблем наступним програмним чином.

Система тимчасово, з обмеженням на певний період часу, зберігає чергу повідомлень для користувача, які були б йому надіслані у разі, якщо він був би онлайн у момент створення повідомлення.

В момент автентифікації користувача у системі, клієнтська частина програмного продукту оформлює підписку на події сервера. Одночасно з цим, система формує повідомлення про те, що відповідний користувач знаходиться онлайн.

Система, після отримання події про зміну статусу користувача на “онлайн”, аналізує наявність черги повідомлень для користувача та надсилає їх у синхронному режимі з використанням сигнального сервера. Запити на підтвердження відеоконференції теж зберігаються у черзі повідомлень.

У випадку, коли користувач системи вже автентифікований у системі та намагається створити відеоконференцію, система перевіряє, чи є відеоконференція за даним ідентифікатором активною.

У разі визначення того, що активної відеоконференції за даним ідентифікатором не знайдено, система забирає дані про розмову (“розмова” є аналогічною до conversation або відеоконференції) із бази даних та створює новий об’єкт активної розмови (відеоконференції) у пам’яті та зберігає усіх потенційних учасників розмови.

Також, система зберігає пари користувачів, що вже перебувають у стані налагодження зв’язку або у стані налагодженого каналу передачі без відповідності до розмов (відеоконференцій). Це необхідно для того, щоб у разі виникнення двох активних відеоконференцій, де присутня однакова пара користувачів, система не намагалася налагодити з’єднання для кожної із розмов. Таким чином, навіть за описаної ситуації, клієнтська частина програмного продукту матиме один канал передачі даних для кожного із учасників відеоконференцій.

Після створення (в оперативній пам’яті) об’єкту активної розмови, система аналізує користувачів системи, що зв’язані із поточною розмовою.

У цьому списку користувачів система фільтрує пари користувачів, що вже мають з’єднання у стані “налагодження з’єднання” або “з’єднано”.

Відфільтрований список учасників відеоконференції можна розділити за станом “онлайн” або “офлайн” на дві групи.

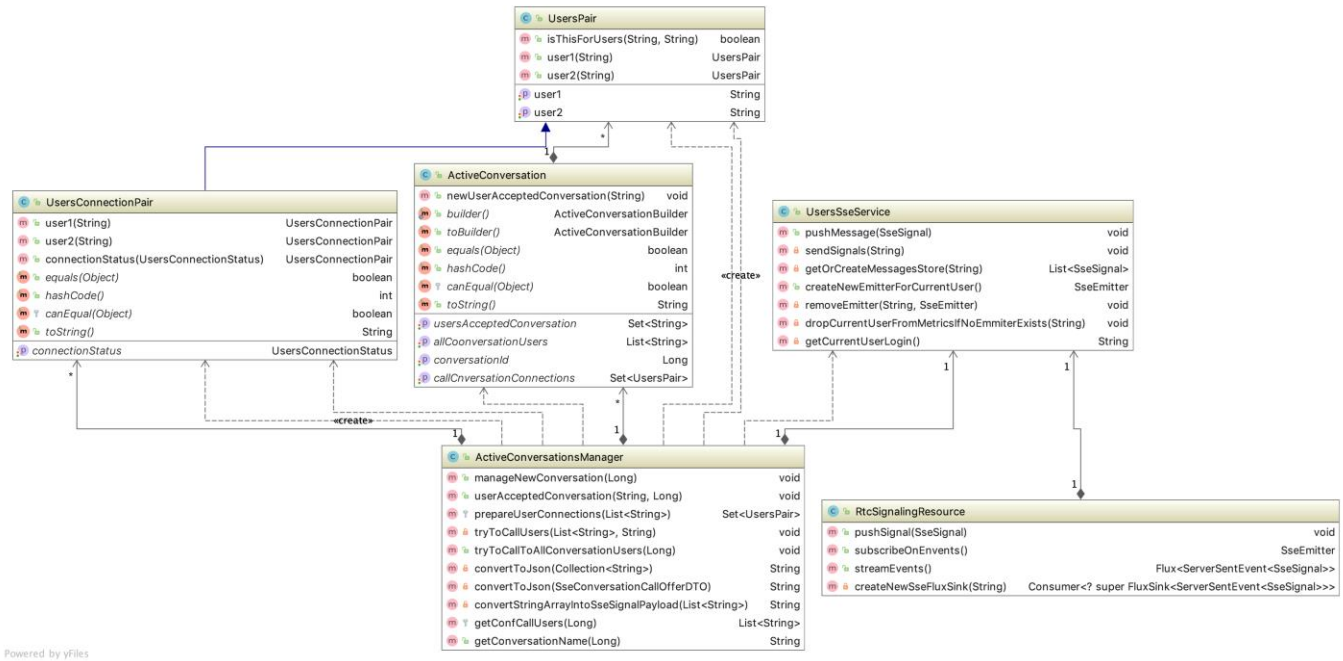

Рисунок 2.6 - Діаграма класів управління відеоконференціями сигнального сервера

Для кожного учасника, що знаходиться у відфільтрованому списку у стані “онлайн”, система, з використанням методів паралелізації, надсилає відповідні повідомлення про підтвердження розмови користувачам.

На рисунку 2.6 проілюстрована діаграма класів компоненту сервера, що відповідає за створення та управління відеоконференцій із прив'язкою до методів сигналізації.

Після вирішення проблем створення сигнального сервера, створення доменних моделей та імплементації логіки управління налаштуваннями та життєвим циклом відеоконференцій, стає можливим налагодження каналу передачі даних з використанням технології WebRTC.

Але для використання даного каналу, клієнтській частині програмного продукту необхідно отримати локальні медіа-потоки (відео та аудіо) з використанням наявних технологій браузера.

Інструментом для цього є Navigator.getUserMedia(), що отримує дозвіл користувача на отримання даних та їх обробку клієнтською частиною. Але обмеженням даного методу є наявність сертифікатів безпеки SSL та префікс визначення HTTP протоколу https.

Для забезпечення цього, було створено самопідписаний сертифікат за допомогою команди (рисунок 2.7)

```
keytool -genkey -alias <your-application> -storetype PKCS12 -keyalg RSA -keysize 2048 -keystore keystore.p12 -validity 3650
```

Рисунок 2.7 - Команда для генерації самопідписаного SSL сертифікату.

Результатом виконання є створення PKCS#12 контейнера із відповідним сховищем ключів. Наступним етапом налаштування є доповнення конфігураційного YML файлу із конфігураціями, вказаними на рисунку 2.8

```
server:
    port: 8080
    ssl:
        key-store: classpath:config/tls/keystore.p12
        key-store-password: password
        key-store-type: PKCS12
        key-alias: foo
        # The ciphers suite enforce the security by deactivating some old and deprecated SSL cipher, this list was tested against SSL Labs (https://www.ssllabs.com
        ciphers: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
```

Рисунок 2.8 - Приклад конфігурації TLS налаштувань YML файлу

Після успішного налаштування схову ключів та відповідних сертифікатів, необхідно передбачити мапу трансформацій запитів з http на https.

Для цього було створено додаткову ланку у фільтрів запиту (рисунок 2.9)

```
import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class HttpsEnforcer
    implements Filter {

    public static final String X_FORWARDED_PROTO = "x-forwarded-proto";
    private FilterConfig filterConfig;

    public void init(FilterConfig filterConfig) { this.filterConfig = filterConfig; }

    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse,
                         FilterChain filterChain) throws IOException, ServletException {

        HttpServletRequest request = (HttpServletRequest) servletRequest;
        HttpServletResponse response = (HttpServletResponse) servletResponse;

        if (request.getHeader(X_FORWARDED_PROTO) != null) {
            if (request.getHeader(X_FORWARDED_PROTO).indexOf("https") != 0) {
                String pathInfo = (request.getPathInfo() != null) ? request.getPathInfo() : "";
                response.sendRedirect( s: "https://" + request.getServerName() + pathInfo);
                return;
            }
        }

        filterChain.doFilter(request, response);
    }
```

Рисунок 2.9 - Приклад конфігурації редіректу запиту з http на https

Таким чином, було реалізовано базисні компоненти серверної частини програмного продукту, що є необхідними для створення відеоконференцій за технологією WebRTC.

Для збереження даних про користувачів та конфігурації учасників відеоконференцій, забезпечення масштабування, конфігурації, зміни та міграції програмного продукту, необхідно реалізувати гнучке налаштування бази даних.

У якості засобу динамічного створення схем та початкових даних баз даних було обрано бібліотеку-фреймворк Liquibase. Всі зміни схем баз даних зберігаються у файлах XML, YAML, JSON і ідентифікуються поєднанням тегу id та author, а також імені самого файлу [37].

Список всіх застосованих змін зберігається в кожній базі даних, над якою проводяться маніпуляції серед з усіх оновлень бази даних, щоб визначити, які зміни необхідно застосувати. Ієрархія файлів liquibase зображено на рисунку 2.10 Liquibase

Після визначення сутностей бази даних, вони агрегуються у файлі master.xml та стають виконуваними (так само, як і файлі змін даних).

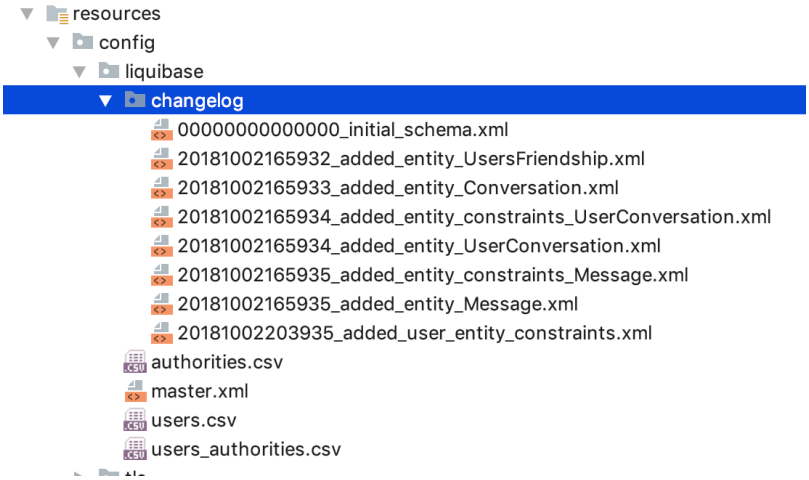

Рисунок 2.10 - Ієрархія файлів змін бази даних Liquibase

За першого запуску сервера, за наявності пустої бази даних, Liquibase асинхронно для основного потоку сервера, перевіряє changelog-и проекту і виконує їх у заданій командою розробки послідовності.

Програмна конфігурація запуску Liquibase з використанням фреймворку Spring може бути реалізовано як показано на рисунку 2.11.

```
@Configuration
public class LiquibaseConfiguration {

    private final Logger log = LoggerFactory.getLogger(LiquibaseConfiguration.class);

    private final Environment env;

    public LiquibaseConfiguration(Environment env) { this.env = env; }

    @Bean
    public SpringLiquibase liquibase(@Qualifier("taskExecutor") TaskExecutor taskExecutor,
            DataSource dataSource, LiquibaseProperties liquibaseProperties) {

        // Use liquibase.integration.spring.SpringLiquibase if you don't want Liquibase to start asynchronously
        SpringLiquibase liquibase = new AsyncSpringLiquibase(taskExecutor, env);
        liquibase.setDataSource(dataSource);
        liquibase.setChangeLog("classpath:config/liquibase/master.xml");
        liquibase.setContexts(liquibaseProperties.getContexts());
        liquibase.setDefaultSchema(liquibaseProperties.getDefaultSchema());
        liquibase.setDropFirst(liquibaseProperties.isDropFirst());
        liquibase.setChangeLogParameters(liquibaseProperties.getParameters());
        if (env.acceptsProfiles(JHipsterConstants.SPRING_PROFILE_NO_LIQUIBASE)) {
            liquibase.setShouldRun(false);
        } else {
            liquibase.setShouldRun(liquibaseProperties.isEnabled());
            log.debug("Configuring Liquibase");
        }
        return liquibase;
    }
}
```

Рисунок 2.11 - Програмна реалізація запуску на виконання Liquibase обробника

Доступ до фізичної бази даних описано у YML конфігурації з використанням фреймворку Spring Data (рисунок 2.12).

```yaml
spring:
    devtools:
        restart:
            enabled: false
        livereload:
            enabled: false
    datasource:
        type: com.zaxxer.hikari.HikariDataSource
        url: jdbc:postgresql://localhost:5432/webrtcHorbenkoProto
        username: postgres
        password: 19069524
        hikari:
            auto-commit: false
    jpa:
        database-platform: io.github.jhipster.domain.util.FixedPostgreSQL82Dialect
        database: POSTGRESQL
        show-sql: false
        properties:
            hibernate.id.new_generator_mappings: true
            hibernate.connection.provider_disables_autocommit: true
            hibernate.cache.use_second_level_cache: false
            hibernate.cache.use_query_cache: false
            hibernate.generate_statistics: false
    liquibase:
        contexts: prod
```

Рисунок 2.12 - Приклад опису конфігурації доступу до БД

Таким чином, було розроблено алгоритм створення відеоконференцій, основні компоненти-обробники даних та налаштування безпеки для використання технології WebRTC. Було побудовано концепт створення, застосування та зміни схеми баз даних та модель persistence-прошарку серверної частини програмного продукту.

3.2. Програмна реалізація клієнтської частини програмного продукту

З огляду на обрану технологію передачі медіа-потоків між користувачами, було обрано концепт Single Page Application та фреймворк Angular 6.2.2, що задовольняє вимогам технології WebRTC.

Основними відповідальностями клієнтської частини програмного продукту є:

1. Створення та управління життєвим циклом каналів передачі даних за технологією WebRTC
2. Управління медіа-потоками локальними та медіа-потоками учасників відеоконференції

3. Відображення медіа-потоків

Після створення каркасу програмного продукту, основних модулів, що відповідають за верстку, модулів, компонентів та сервісів системи, що відповідають за налаштування безпеки та роутингу між сторінками, пропонується звернути увагу на створення компонентів, що відповідають за сигналізацію подій сервер-клієнт та клієнт-сервер, оскільки налаштування створення каналів передачі даних за WebRTC буде виконано саме з використанням цих компонентів.

Передача сигналів клієнт-сервер виконується з використанням @angular/common/http/ HttpClient класу, що інтегровано в якості модуля фреймворку Angular. Дані сигнали передаються на сервер за HTTP протоколом з відповідністю концепту REST.

Більш складною є імплементація прийому сигналів із сервера. Було попередньо обрано протокол передачі даних Server-Sent Events у якості каналу передачі подій сервера. Клієнтською частиною програмного продукту створюється підписка на події сервера, що повинна існувати постійно (для одної вкладки браузера). Підписка повинна створюватись після успішної автентифікації системи та знищуватись після операції logout.

Проблема може виникнути у разі збереження даних cookie барузером. У цьому випадку, після створення вкладки браузеру та переходу, наприклад, на сторінку відеоконференцій, ініційовану з історії, повна автентифікація виконана не буде. Для коректності роботи системи, підписка на події сервера повинна відбуватися після перевірки двох тригерів:

1. Клієнт автентифікувався у системі;
2. Веб-сторінка розгорнута браузером

Для обробки автентифікації було створено об’єкт Subject<T> модуля rxjs та створення об’єкта типу Observable<T> у сервісі автентифікації клієнтської частини програмного продукту.

Для обробки другого тригеру (веб-сторінка розгорнута браузером), було проаналізовано програмний компонент на наявність постійних HTML-шаблонів та відповідних до представлення компонентів.

Визначено, що модуль верхнього меню (navbar.component, navbar.module, navbar.html та інші) фізично розгортуються один раз у розрізі життєвого циклу існування веб-частини програмного продукту в одному вкладенні браузеру.

Фреймворк Angular передбачає додаткові методи життєвого циклу компонентів – Anngular Lifecycle Hooks. Потребує уваги hook ngOnInit, тіло якого викликається в процесі розгортки компоненту та може бути реалізовано розробниками.

Саме під час розгортки модулю navbar, виконується надсилання події про розгортку сторінки, підписчиком якої є сервіс SseSignalingService, який, у свою чергу, виконує перевірку наявності підписки на події сервера та стану автентифікації користувача.

SseSignalingService є класом та сервісом у термінології Angular. Це гарантує існування одного об’єкту даного класу та можливість ін’єкції цього сервісу в усі необхідні модулі, компоненти та сервіси клієнтської частини програмного продукту.

Завдяки описаній вище реалізації, отримано гарантію створення підписки на події сервера навіть за умови відкриття дублікату вкладки браузеру, фунцкії remember me та збереження cookie браузером.

Саме сервіс SseSignalingService відповідає за сигналізацію клієнт-сервер та сервер-клієнт. Сервіс створює об’єкт типу Subject<SseMessage> та об’єкт типу Observable<SseMessage> із об’єкта типу Subject (рисунок 2.13).

```
private newSseSignalingMessageSubject = new Subject<SseMessage>();
newSseSignalingMessageObservable$ = this.newSseSignalingMessageSubject.asObservable();
```

Рисунок 2.13 - Створення об’єктів для можливості реалізації шаблону GOF Observer

Ці два об’єкти реалізують шаблон проектування Observable, завдяки чому компоненти системи можуть провести ін’єкцію сервісу та оформити підписку на отримання подій сервера, прийняті каналом SSE.

Таким чином, було реалізовано логіку сигналізації подій та надано можливість підписки на нові сигнали із сервера та інструмент надсилання подій клієнта на сервер.

Наступним етапом розробки є реалізація логіки створення та управління каналами передачі даних за протоколом WebRTC (рисунок 2.14).

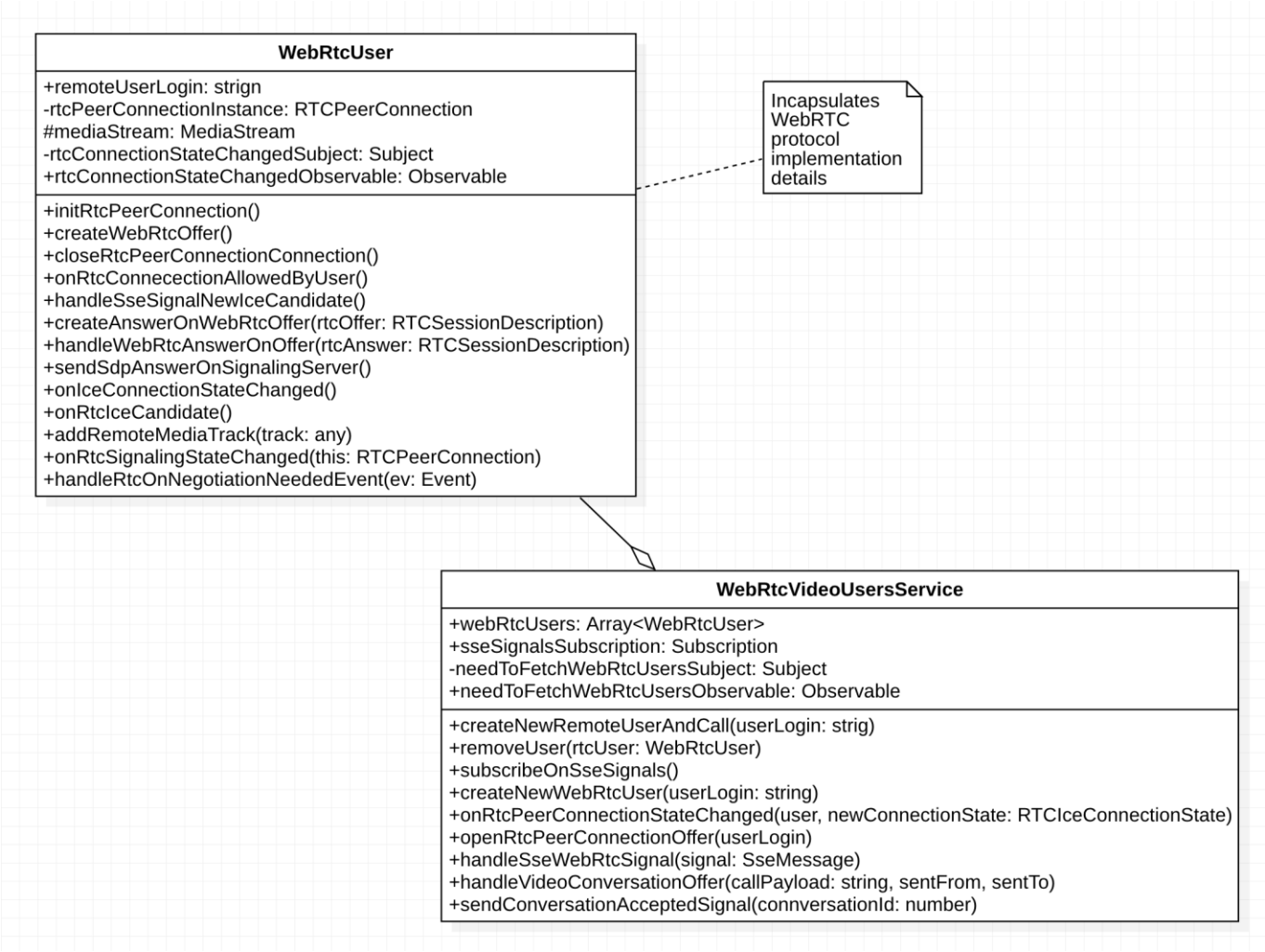

Рисунок 2.14 - Діаграма класів та сервісів реалізації логіки управління каналами передачі даних за WebRTC

Дані два класи інкапсулюють логіку створення та управління учасників розмов, каналів передачі даних за технологією WebRTC (за описом у розділах 1.3, 2.2, 2.3) та збереження медіа-потоків користувачів.

Важливо акцентувати увагу на тому, що WebRtcVideoService зберігає метадані та потоки передачі даних, тому даний клас повинен бути сервісом (Service) у термінології фреймворку Angular.

Після реалізації функціональності створення та управління учасників розмови (відеоконференції), необхідно реалізувати логіку відображення медіа-потоків

учасників розмови на інтерфейсі користувача клієнтської частини програмного продукту.

Одним із основних факторів побудови відображення клієнтської частини програмного продукту є підтримка цілісності інформації, що надається користувачеві. Основною інформацією для відображення програмного продукту є медіа-потоки учасників відеоконференції (блоки відео для кожного учасника розмови) та спливаючі повідомлення про підтвердження запиту на дозвіл користувача в участі у відеоконференції.

Розроблена архітектура передбачає виконання основної логіки налаштування відео зв'язку у «фоновому» режимі без участі клієнта як такого. Також, повідомлення на підтвердження повинні бути відображені користувачу без прив'язки до конкретної сторінки системи, де наразі знаходиться користувач та без необхідності до дій клієнта.

Розглянемо детальніше концепт відображення спливаючих вікон підтвердження. Повідомлення про підтвердження створюються в момент обробки повідомлення-сигналу від сигнального сервера. Після обробки інформації, формується модель відображення, що повинна бути відображена на сторінці користувача.

Було вирішено створити компонент-деталь, що відповідає за події клієнтської частини програмного продукту, що пов'язана з відповідним шаблоном відображення моделей. Даний компонент, у свою чергу, з використанням механізму ін'єкції HTML-шаблонів, агрегується у компоненті-мастері відображення стеку повідомлень.

Шаблон відображення мастер-компоненту, у свою чергу, агрегується компонентом основної сторінки відображення. Цей підхід дозволяє відображати повідомлення на будь-якій активній сторінці веб-частини продукту.

Але оновлення моделей відображення фреймворком Angular стандартно відбувається за зміни моделей, що зв'язані з відображенням конкретної сторінки. Таким чином, ре-рендеринг компонентів веб-сторінки, за життєвим циклом веб-частини, потребує дій користувача (наприклад, подію кліку).

Але можлива ситуація, за якою користувач не ініціює події браузеру і лиш чекає на повідомлення про підтвердження запиту на створення розмови.

Використання методу ngDoCheck() не дало б бажаного результату, оскільки компонент змінює значення внутрішніх об'єктів масиву. Так як посилання на модель об'єкту масиву не змінюється, Angular Lifecycle Hook ngDoCheck() не гарантує збереження консистентного стану даних-значень.

Вирішенням даної проблеми є ін'єкція об'єкту NgZone та виконання дій всередині області видимості зони перевірки моделі Angular. Приклад використання наведено на рисунку 2.15.

```
this.ngZone.run( fn: () => {
    this.webRtcUsersManager.sendConversationAcceptedSignal(this.alertData.conversationId
    this.alertsManagerService.closeAlert(this.alertData);
});
```

Рисунок 2.15 - Використання NgZone для забезпечення гарантованої перевірки моделей

Таким чином, повідомлення буде відображено користувачеві веб-продукту без необхідності додаткових дій або оновлення сторінки браузеру.

Також, одним із важливих аспектів побудови архітектури відображення є області відображення відео-потоків учасників програмного продукту. Вище згадані компоненти, аналогічно до повідомлень-запитів, побудовано за принципом майстер-деталь. Сторінка відеоконференції агрегує в собі компонент відображення учасників відеоконференцій (майстер), що, у свою чергу, агрегує компоненти відображення відео-потоків для кожного користувача (деталі). Важливою особливістю розробки є те, що користувач має можливість до переходу на різні сторінки клієнтської частини програмного продукту.

Кожний перехід ініціює знищення компонентів відображення та деталей фреймів відео-потоків. При зворотному переході на сторінку, компоненти створюються заново та відбувається рендеринг компонентів сторінки заново. Так, як локальні медіа потоки та медіа-потоки користувачів інкапсюльовано в Angular-сервісах, проблем із перетвореннями медіа-потоків бути не повинно. У свою чергу,

виникає потреба до відображення відео-потоків користувачів, що тільки-но були успішно з’єднані.

Для цього було використано концепт, схожий на відображення подій-. Рендернинг нових елементів відео відбувається тільки за умови зміни статусу передачі даних на connected, при чому, додання нового об’єкту шаблону відображення медіа-потоків у мастер-компонент відеоконференцій буде виконано з використанням зони дії потоку Angular.

Таким чином, навіть за повної бездіяльності клієнта системи, актуальні елементи відображення медіа-потоків та важливих повідомлень будуть відображені без необхідності додаткових маніпуляцій з боку клієнта програмного продукту.

3.3. Висновки до розділу 3

Було програмно реалізовано серверний програмний компонент та базисну бізнес-логіку створення та управління сигналізації, що виступає у ролі сигнального сервера, необхідного для реалізації технології передачі медіа-потоків WebRTC.

Реалізовано логіку створення та управління відеоконференціями, програмно реалізовано прошарки серверної частини програмного продукту: прошарок ресурсів, бізнес-логіки та доменів бази даних з використанням фреймворку Spring. Також реалізовано логіку зміни схеми бази даних та даних з використанням Liquibase.

Визначені ключові місця появи колізій на клієнтській частині програмного продукту та розроблено методи авторизації та автентифікації, написано реалізацію компонентів обміну сигналами із сигнальним сервером, програмно реалізовано створення відеоконференцій за підтвердженням та відповідна модель компонентів системи.

4. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Протягом останніх років розвиток автоматизованих інформаційних систем став поштовхом до розвитку соціальних мереж.

Ідея проекту полягає у створенні програмного забезпечення, що надає можливість до створення відео конференцій між користувачами. Програмний продукт орієнтовано на використання у навчальних закладах різних типів акредитації в якості інструменту для проведення дистанційних занять.

Завдання розділу полягає у визначенні перспективності запропонованих у цій роботі науково-технічних рішень і пропозицій за допомогою маркетингового аналізу та оцінці можливостей їх ринкового впровадження [38].

В розділі надається опис ідеї стартап-проекту та визначаються основні напрями до його використання програмного продукту та технічні відмінності від конкурентних продуктів.

За даними цього розділу можна визначити доцільність переходу до наступних етапів реалізації проекту, що розроблюється.

4.1. Опис ідеї проекту

Ідея стартап-проекту є основним чинником: що впливає на успішність розвитку стартап-проекту. Саме тому чітке визначення ідеї та можливих напрямів її застосування є дуже важливим етапу реалізації продукту.

У вигляді таблиці 4.1 надається опис базисної ідеї проекту, надається цілісне уявлення про зміст ідеї та потенційні ринки реалізації продукту.

Таблиця 4.1. Опис ідеї стартап-проекту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Створення клієнт-серверного програмного продукту для забезпечення відеоконференцій	1. Проведення дистанційних лекційних занять	Проведення відеоконференцій без обмеження на кількість користувачів
	2. Використання продукту в якості соціальної мережі	Клієнт-серверний програмний продукт може бути використаний лише за наявності браузера на пристрої

Після визначення основної ідеї стартап-проекту необхідно визначити сильні та слабкі сторони проекту, що розробляється. Для цього було визначено найбільш значимі техніко-економічні характеристики ідеї, коло можливих конкурентів проекту, продуктів-аналогів та продуктів-замінників, що існують на ринку на момент написання роботи (таблиця 4.2).

Таблиця 4.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	**Техніко-економічні характеристики ідеї**	**(потенційні) товари/концепції конкурентів**				**W**	**N**	**S**
		Проект, що розроблюється	**FaceTime**	**Skype**	**Skype WEB**			
1	Технологія передачі даних	WebRTC	TCP+ICE	WebSocket	WebSocket			+

2	Макимальна кількість учасників відеоконференції	64 (максимальна технічна - 256)	32	25	2			+
3	Середовища роботи	Що підтримують Google Chrome/Mozilla Firefox	IOS, MacOS	Windows, MacOS, IOS, Android	Що підтримують Google Chrome/Mozilla Firefox		+	
4	Залежність від наявності сервера	Тільки на етапі встановлення з'єднання	Повна	Повна	Повна			+

Як можна побачити з таблиці 4.2 , сильними сторонами програмного продукту, що розробляється, є кількість учасників відеоконференції, гнучкість використання без необхідності до встановлення додаткового програмного забезпечення та залежність від серверної частини програмного забезпечення тільки на етапі встановлення з'єднань.

Тому актуальною є проблема розробки програмного продукту, що не потребує встановлення інфраструктури для підтримки серверної частини програмного забезпечення.

4.2. Технологічний аудит ідеї проекту

Для того, щоб ідеї були якісно реалізовано, важливим є вибір технологій реалізації. Саме тому після визначення ідеї проекту наступним кроком аналізу можливостей імплементації стартап-проекту є визначення технологій її реалізації.

В таблиці 4.3 наведено аналіз технологічної здійсненності проекту.

Таблиця 4.3. Технологічна здійсненність ідеї проекту

Ідея проекту	**Технології її реалізації**	**Наявність технологій**	**Доступність технологій**
Створення клієнт-серверного програмного продукту для забезпечення відеоконференцій	Мови програмування Java, JavaScript, технологія HTML 5	✓	Доступно
	Середовище розробки Itellij IDEA	✓	Доступно

З таблиці 4.3 можемо зробити висновок, що усі необхідні технології доступні для використання та були обрані для подальшої імплементації ідеї стартап-проекту.

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів [38].

Попередня характеристика потенційного ринку стартап-проекту наведена в таблиці 4.4.

Таблиця 4.4. Попередня характеристика потенційного ринку стартап-проекту

№ п/п	**Показники стану ринку (найменування)**	**Характеристика**
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	5000
3	Динаміка ринку (якісна оцінка)	Зростає

4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні для ПЗ
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	41%

Вкладати гроші в проект може бути вигідно тільки, якщо середня норма рентабельності в галузі є більшою ніж банківські відсотки на вкладення. Як бачимо з таблиці 4.4, продукт є привабливим для входження на ринок за попереднім оцінюванням.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 4.5).

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають.

Характеристики потенційних клієнтів стартап-проекту наведено в таблиці 4.5.

Таблиця 4.5. Характеристика потенційних клієнтів стартап-проекту

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1	Необхідність проведення відеоконференцій більше ніш з 30ма користувачами	Навчальні заклади (забезпечення можливості дистанційного навчання)	В залежності від ціни та відповідності потребам клієнтів	Швидкодія, безпека передачі особистих даних

		Користувачі соціальних мереж	В залежності від відповідності потребам клієнтів	Швидкодія, безпека передачі особистих даних
2	Кросплатформен-ність	Користувачі соціальних мереж	В залежності від відповідності потребам підприємства	Швидкодія, безпека передачі особистих даних, можливість доступу з різних пристроїв

Важливим у розробці стартап-проекту є попереднє визначення загроз, що можуть перешкодити його імплементації та успішному розвитку в обраній сфері.

В таблиці 4.6 наведено основі фактори загроз та можливі дії, що призведуть до мінімізації негативного впливу цих факторів на розвиток проекту. Фактори в таблиці 4.6 подані в порядку зменшення значущості.

Таблиця 4.6. Фактори загроз

№ п/п	**Фактор**	**Зміст загрози**	**Можлива реакція компанії**
1	Інвестиції	Відсутність інвестицій	- Венчурні інвестиції; - Краудфандинг
2	Конкуренція	Вихід на ринок, де вже існують великі компанії	-Запропонувати поглинання компанії - Розробка унікальної функціональності
3	Довіра користувача	Новому продукту на ринку важко	- Розробка маркетингових концепцій;

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
		завоювати довіру користувача, оскільки перевага, зазвичай, дістається перевіреним брендам	- Надання безкоштовного обмеженої функціональності продукту користувачам
4	Науковий	Розробка технологій передачі медіа-потоків між користувачами	- Проведення власних подальших досліджень в цій галузі; - Паралельна імплементація та впровадження нових технологій

Не менш значущим етапом для успішної реалізації стартап-проекту, є визначення факторів, що сприяють ринковому впровадження продукту.

В таблиці 4.7 представлені фактори можливостей та можлива реакція компанії на них (фактори подані в порядку зменшення значущості).

Таблиця 4.7. Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Наукова новизна	Використання новітньої технології передачі медіа-даних між браузерами	При виході на ринок, саме ця характеристика буде вигідно відрізняти

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
			цей продукт від конкурентів
2	Наявність попиту	Попит на відеоконференції з можливістю одночасного підключення більше 32х користувачів	Підтримка програмного забезпечення
3	Економічний	Відсутність необхідності підтримки та масштабування серверної частини програмного продукту	Оптимізація використання ресурсів серверів

Як бачимо з таблиць 4.6 та 4.7 в проекті присутня наукова новизна у вигляді використання новітньої технології, що є основним позитивним фактором для виходу на ринок.

Подальшим кроком в проведенні маркетингового аналізу, після визначення загроз та можливостей, є проведення ступеневого аналізу конкуренції на ринку для визначення її загальних рис (таблиця 4.8).

Таблиця 4.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції: монополістична	Фірми конкурентів Apple та Microsoft	Впровадження новітніх технологій Регуляції цін на корпоративне використання Реклама

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)**
За рівнем конкурентної боротьби: - міжнародна	Фірми-конкуренти є міжнародними	Адаптація програмного засобу для зарубіжних клієнтів
За галузевою ознакою: - міжгалузева	Продукт придатний для використання у різних галузях	Цільова адаптація програмного засобу для зарубіжних клієнтів
Конкуренція за видами товарів: - товарно-видова	Вид товару — програмне забезпечення	Програмне забезпечення, що створюється, повинне враховувати сильні та слабкі сторони конкурентів
За характером конкурентних переваг: - нецінова	Основними перевагами цього продукту є наявність характеристик відсутніх у конкурентів	Підтримка якості за рахунок постійного вдосконалення
За інтенсивністю: - марочна	Бренди існують та конкурують	Просування бренду, реклама, яка вказує на переваги саме цього продукту

Після аналізу конкуренції на ринку необхідно провести більш детальний аналіз умов конкуренції в галузі. Аналіз проводився за моделлю 5 сил М. Портера (таблиця 4.9[38].

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 4.9) - за моделлю п’яти сил М. Портера, яка вирізняє

п’ять основних факторів, що впливають на привабливість вибору ринку з огляду на характер конкуренції:

1. Конкурент, що вже є в галузі.
2. Потенційні конкуренти.
3. Наявність товарів замінників.
4. Постачальники, що конкурують за ринкову владу.
5. Споживачі.

Сильні позиції компанії по кожному фактору забезпечують їй більший вплив на ринок. Аналіз складових цієї моделі є основою для розроблення переліку факторів конкурентоспроможності для певного обраного ринку.

Таблиця 4.9. Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	**Потенційні конкуренти**	**Постачальники**	**Клієнти**	**Товари-замінники**
Складові аналізу	FaceTime, Skype	Наявність вже існуючих рішень	-	Контроль якості продукту	Агресивна рекламна кампанія, імплементація новітніх технологій
Висновки:	Конкуренти уже завоювали довіру провідних компаній - замовників	Є можливості виходу на ринок, але є і конкуренти. Строки – 12 місяців.	-	Клієнти диктують усі умови роботи на ринку	Оптимізація технічної бази та імплеменація новітніх технологій

За результатами аналізу таблиці 4.9 можна зробити висновок, що компанія має можливість забезпечити необхідні темпи обороту капіталу та здатна впливати на інших учасників ринку. Тож проект має позитивні перспективи на ринку, що розглядається.

На основі аналізу конкуренції (таблиця 4.9), ураховуючи характеристики ідеї проекту (таблиця 4.2), вимоги користувачів до продукту (таблиця 4.5) та факторів загроз (таблиця 4.6) і можливостей (таблиця 4.7) було визначено перелік факторів конкурентоспроможності та обґрунтовано їх вибір шляхом наведення чинників, що роблять їх значущими (таблиця 4.10). актори наведені в порядку спадання значущості.

Таблиця 4.10. Обґрунтування факторів конкурентоспроможності

№ п/п	**Фактор конкурентоспроможності**	**Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)**
1	Швидкодія	Відсутність необхідності постійної трансформації даних та трансформацій на серверній стороні програмного забезпечення надають більший показник швидкодії, ніж у прямих конкурентів
2	Надійність та якість зв'язку	Відсутність серверної частини програмного продукту у ланцюгу передачі медіа-потоків між браузерами підвищують надійність та якість відео- та аудіо- потоків навіть за технічної відмови сервера
3	Зручність інтерфейсу користувача	Веб-інтерфейс направлений на користувача та не потребує встановлення сторонніх додатків

Отже, як видно з таблиці 4.10, можна виділити три основних фактори, що впливають на конкурентоспроможність компанії. Важливим є обґрунтування факторів конкурентоспроможності для подальшого аналізу. На вибір цих факторів впливали такі чинники, як значущість для споживача та наявність можливості для подальшого розвитку самого продукту.

За визначеними у таблиці 4.10 факторами конкурентоспроможності проведено порівняльний аналіз сильних та слабких сторін розроблюваного стартап-проекту. Надана оцінка в балах (від 1 до 20) по кожному фактору та рейтинг товарів-конкурентів у порівнянні з розробленим продуктом.

Таблиця 4.11. Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	**Фактор конкурентоспроможності**	**Бали 1-20**	**Рейтинг товарів-конкурентів у порівнянні з розробленим продуктом**						
			-3	**-2**	**-1**	**0**	**+1**	**+2**	**+3**
1	Швидкодія	19		+					
2	Надійність та якість зв'язку	19	+						
3	Зручність інтерфейсу користувача	19				+			

З таблиці 4.11, можемо зробити висновок, що сильними сторонами продукту, що розробляється, є швидкодія, надійність та якість зв'язку та зручність інтерфейсу користувача.

Узагальнюючи результати попередньо проведеного аналізу сильних та слабких сторін стартап-проекту (таблиця 4.11), переліку ринкових загроз (таблиця 4.6) та можливостей (таблиця 4.7), здійснено узагальнюючий SWOT-аналіз стартап-проекту (таблиця 4.12) для визначення повної картини стану продукту на ринку.

Таблиця 4.12. SWOT- аналіз стартап-проекту

Сильні сторони:	**Слабкі сторони:**
- Швидкодія - Надійність - Зручність користувацького інтерфейсу - Часткова незалежність від серверної частини програмного продукту	- Відсутність бренду - Відсутність репутації серед клієнтів
Можливості:	**Загрози:**
- Вдосконалення інтерфейсу користувача - Оптимізація клієнтської частини програмного продукту - Розробка нової функціональності для соціальних мереж - Розробка інтеграції з іншими інформаційними системами - Просування бренду	- Відсутність інвестицій - Висока конкуренція

За результатами SWOT-аналізу для виведення стартап-проекту на ринок розроблено ряд альтернатив ринкової поведінки, що до впровадження стартап-проекту (таблиця 4.13).

Таблиця 4.13. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Залучення інвестицій (розробка програмного продукту, реклама, просування бренду)	60%	12 місяців
2	Інтеграція (співпраця) із конкурентами	70%	36 місяців
3	Гнучка система лояльності для корпоративних клієнтів	80%	5 місяців

Альтернативи обиралися з оглядом на строки реалізації проекту, ймовірність отримання ресурсів для запуску стартапу та потенційні проекти конкурентів, що також можуть бути виведені на ринок у найближчий час (таблиця 4.9).

4.4. Розроблення ринкової стратегії стартап-проекту

Важливим етапом маркетингового аналізу є розроблення ринкової стратегії проекту. Під час цього етапу визначаються ринкові цілі, та визначаються оптимальні можливості поширення на ринку.

Для початку визначаються цільові групи потенційних споживачів (таблиця 4.14), що робить можливим вибір стратегії охоплення ринку, визначення орієнтованих споживачів.

Таблиця 4.14. Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Навчальні заклади	Високий	Високий	Високий	Попит на продукт є високим, проте можуть виникнути складнощі з високим рівнем конкуренції
2	Користувачі соціальних мереж	Високий	Середній	Високий	Готовність споживачів сприйняти продукт є високим, проте проте можуть виникнути складнощі з високим рівнем конкуренції
Обрано першу цільову групу, вона є більш перспективною, проте друга цільова група теж потребує уваги.					

Як видно з таблиці 4.14 до розгляду беруться дві основні цільові групи потенційних клієнтів. Простота входу в обидва сегменти є досить низькою, оскільки в першому випадку заважають конкуренти, а в другому може бути не достатній попит. Обрано першу цільову групу, вона є більш перспективною, оскільки, якщо продукт

компанії все ж вийде на ринок, ймовірність того, що він окупиться є досить високою. Проте друга цільова група теж потребує уваги.

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку, яка визначається у таблиці 4.15.

Таблиця 4.15. Визначення базової стратегії розвитку

№ п/п	**Обрана альтернатива розвитку проекту**	**Стратегія охоплення ринку**	**Ключові конкурентоспроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку***
1	Розробка та підтримка програмного продукту	Ринкове позиціонування	Швидкодія передачі медіа-потоків та надійність серверної частини програмного продукту	Стратегія диференціації

Далі, на основі базової стратегії розвитку, надається визначення стратегії конкурентної поведінки (таблиця 4.16).

В даному випадку обрано стратегію наслідування лідерам. Компанії, що приймають слідування за лідером – це підприємства з невеликою часткою ринку, які вибирають адаптивну лінію поведінки на ринку, усвідомлюють своє місце на ньому і йдуть у фарватері фірм-лідерів. Головна перевага такої стратегії – економія фінансових ресурсів, пов’язаних з необхідністю розширення ринку, постійними інноваціями, витратами на утримання домінуючого положення.

Таблиця 4.16. Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні	Так	Ні	Стратегія наслідування лідерам

Наступним кроком в дослідженні ринку є вибір стратегії її позиціонування перед обраною цільовою групою потенційних клієнтів (таблиця 4.17).

Таблиця 4.17. Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	– Швидкодія передачі медіа-потоків; – Кількість учасників відеоконференції;	Стратегія диференціації	Імплементація нової технології передачі медіа-даних, що дозволяє досягнути підвищенню	– швидкість; – надійність; – якість:

	– Надійність серверної частини програмного продукту		швидкості та якості з'єднання	

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (таблиця 4.5), а також в залежності від обраної базової стратегії розвитку (таблиця 4.15) та стратегії конкурентної поведінки (таблиця 4.16) розроблено стратегію позиціонування (таблиця 4.17). що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати проект стартап-проекту.

4.5. Розроблення маркетингової програми стартап-проекту

Маркетингова програма стартап-проекту є одним із основних чинників росту кількості користувачів системи.

Завершальним етапом у проведенні маркетингового аналізу стартап-проекту є розроблення ефективної стратегія ринкового впровадження потенційного товару на базі аналізу ринкового середовища, розроблення маркетингової програми стартап-проекту.

Першим кроком до розробки маркетингової стратегії є формування маркетингової концепції товару, який отримає споживач.

В таблиці 4.18 підсумовані результати попереднього аналізу конкурентоспроможності товару з врахуванням попередньо обраних трьох основних асоціацій з товаром, що дозволить сформувати маркетингову концепцію товару, який отримає клієнт.

Таблиця 4.18. Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Велика кількість учасників відеоконференції	Забезпечення 32+ учасників відеоконференції одночасно	Найближчий конкурент (FaceTime) пропонує лише 32 учасники відеоконференцій
2	Надійність серверної частини програмного продукту	Використання новітньої технології передачі медіа-потоків	Обрана технологія використовує серверну частину лише для налаштування зв'язку
3	Швидкодія	Технологія передачі даних між браузерами не потребує додаткової обробки на серверній частині програмного продукту	Зменшення навантаження з серверної частини програмного продукту

В таблиці 4.19 наведений опис трирівневої моделі товару з уточненням ідеї продукту, його фізичних складових та особливостей процесу його надання.

Таблиця 4.19. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Програмний продукт для забезпечення відеоконференцій з багатьма учасниками із найменшим навантаженням на серверну частину програмного забезпечення		
	Властивості/характеристики	М/Нм	$В_р/Т_х/Т_л/Е/О_р$
	1. Швидкодія	-	-

II. Товар у реальному виконанні	2. Урахування сплеску температур 3. Користувацький інтерфейс 4. Універсальність програмного засобу 5. Швидкість моделювання		
	Якість: згідно до стандарту ISO 29119 буде проведено тестування		
	Пакування: CD диск		
	Марка: Марка: O.Horbenko. ConfCall.2		
III. Товар з підкріпленням	До продажу: базова версія		
	Після продажу: постійна підтримка та вдосконалення		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності			

Для проведення маркетингового аналізу важливим фактором є визначення рівня цін на товари конкурентів. Далі наведено визначення цінових меж (таблиця 4.20), якими необхідно буде керуватися при встановленні ціни на потенційний товар, включаючи аналіз рівня доходів цільової групи споживачів та рівень цін на товари-аналоги та товари-замінники.

Таблиця 4.20. Визначення меж встановлення ціни

№ п/п	**Рівень цін на товари замінники (грн/міс)**	**Рівень цін на товари аналоги (грн/міс)**	**Рівень доходів цільової групи споживачів (грн/міс)**	**Верхня та нижня межі встановлення ціни на товар/послугу**
1	100	200	50000	400 – 5000

В таблиці 4.21 відображена оптимальна система збуту товару. Як результат, було прийнято рішення: доцільно користуватися одноріневим каналом збуту.

Таблиця 4.21. Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Покупка програмного забезпечення	Продаж	– 0(напряму) – 1(через одного посередника)	Напряму або через одного посередника

Концепція маркетингових комунікацій (таблиця 4.22) є останньою складовою маркетингової програми. Ця концепція бере за основу попередньо обране рішення для позиціонування та визначення специфіки поведінки споживачів.

Таблиця 4.22. Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Клієнти обирають у перевірених фірм	Інтернет	– швидкість; – якість; – надійність	Ознайомлення користувача з перевагами цього продукту	Інтернет, агресивна реклама

Як бачимо в цьому пункті сформовано ринкову програму, що включає в себе концепцію збуту та просування товару, а також попередній аналіз можливостей ціноутворення, що спирається на потреби та можливості потенційних клієнтів компанії, а також відповідну обрану альтернативу ринкової поведінки для найбільш ефективного виходу розроблюваного продукту на ринок.

4.6. Висновки до розділу 4

В цьому розділі було проведено аналіз першого етапу розроблення стартап-проекту, а саме — висвітлено маркетингові аспекти створення стартап-проекту. Можна відзначити, що можливість ринкової комерціалізації проекту є досить високою завдяки попиту на продукт на ринку програмного забезпечення для дистанційного навчання.

Для проекту обрано дві основні групи користувачів: навчальні заклади та користувачі соціальних мереж. Стратегія диференціації обрана як базова стратегія розвитку.

Новому продукту на ринку важко завоювати довіру користувача, оскільки перевага, зазвичай, дістається перевіреним брендам. А тому, для виходу на великий ринок необхідно буде вкласти великі кошти для реклами та просування свого бренду.

На ринку існує монополістична конкуренція, є декілька фірм конкурентів, тому вихід на ринок передбачає складнощі.

Проте, проект має свої сильні сторони, зокрема однією із основних є те, що програмний продукт надає можливість створення відеоконференцій з більш ніж тридцятьма користувачами без програшу в якості відео та аудіо. Найімовірніша альтернатива впровадження стартап-проекту — інтеграція (співпраця) з конкурентами.

З огляду на це, можна зробити висновок, що подальша імплементація проекту є доцільною.

ВИСНОВКИ

1. У роботі проведено аналіз сучасних технік та протоколів передачі медіа-потоків та обґрунтовано вибір новітньої технології WebRTC, крім того, для передачі конфігураційних даних між учасниками відеоконференції за протоколом WebRTC, пропонується використання технології Server-Sent Events.

2. Виконано проектування загальної архітектури програмного продукту, що передбачає використання WebRTC та дозволяє зменшити навантаження на сервер без втрати якості відео та аудіо потоків.

3. Обґрунтовано використання принципу Single Page Application для клієнтської частини програмного продукту, що не допускає перезавантаження сторінки браузера і, таким чином забезпечує стабільний зв'язок користувачів, кількість яких обмежується технічними характеристиками браузера.

4. Розроблено клієнт-серверний програмний продукт для проведення відеоконференцій, що включає серверний програмний компонент, який реалізує бізнес-логіку створення та управління конфігураціями відеоконференцій, та клієнтську частину на основі SPA та WebRTC, який на відміну від існуючих аналогів дозволяє одночасну передачу медіа-потоків між користувачами кількістю до 256. За рахунок використання технології WebRTC вдалося позбутися необхідності великих потужностей серверів, що дозволяє зменшити грошові витрати.

5. Проведено аудит ідеї проекту та визначено факт доцільності подальшої реалізації проекту. Зроблено аналіз ринкових можливостей запуску стартап-проекту, розроблено ринкову стратегію та маркетингову програму.

В якості подальшого розвитку проекту рекомендується вдосконалення інтерфейсу користувача, що зробить продукт більш привабливим для потенційних замовників, а також надання можливості інтеграції з іншими продуктами, наприклад: системою дистанційного навчання НТУУ «КПІ ім. Сікорського».

СПИСКОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Loreto S. Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP. [Електронний Ресурс]. / G. Wilkins, S. Salsano, P. Saint-Andre. Режим доступу до ресурсу: Available: https://tools.ietf.org/html/ rfc6202#page-16.

2. Russell, Alex. "Comet: Low latency data for the browser." / Alex Russell [Електронний Ресурс] Режим доступу до ресурсу: http://alex.dojotoolkit.org/?p=545, 2006.

3. Горбенко О.Ю. Огляд технології WebRTC для реалізації програмного забезпечення відеоконференцій / Горбенко О.Ю., Третяк В.А. / Сталий розвиток – XXI століття: управління, технології, моделі. Дискусії – 2018: колективна монографія / Міненко М.А. та ін.. НТУУ КПІ ім. Ігоря Сікорського, Національний університет "Києво-Могилянська академія"; Вища економіко-гуманітарна школа. – Київ, 2018 . – С.467-472.

4. Estep, Eliot. Mobile html5: Eciency and performance of websockets and server-sent events. / Estep, Eliot. [Електронний Ресурс] Master thesis, 3.3 Web techniques, June 2013. Режим доступу до ресурсу: URL http://goo.gl/n0TTHo.

5. Hickson, I. Server-sent events. W3C Working Draft WD-eventsource-20091222, / I. Hickson, [Електронний Ресурс] Режим доступу до ресурсу: http://www. w3. org/TR/eventsource, page 18.

6. RFC-6455. – Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011. Режим доступу до ресурсу: https://tools.ietf.org/html/rfc6455

7. Furukawa, Y. Web-based control application using WebSocket. / Y. Furukawa, [Електронний Ресурс] European Synchrotron Radiation Facility ESRF, 38 Grenoble (France); 1423 p;ISSN 2226-0358; Worldcat; 2012; p. 695-697. Режим доступу до ресурсу : http://www.iaea.org/INIS/contacts

8. Liu, Q. Research of web real-time communication based on web socket. / Liu, Q., Sun, X. // International Journal of Communications, Network and System Sciences, 5(2012), p. 797–801.

9. Crowther, R. HTML5 in Action. Manning / Crowther, R., Lennon, J., Blue, A., Wanish, G., Heilmann, C. / Publisher: Manning Publications Release – 2014. — 444 p.

10. Беседа Д.Г. Исследование технологии построения приложений реального времени с использованием протокола websocket / Беседа Д.Г., Семенистый Н.В., Аноприенко А.Я. // Конференция ИУС КМ - 2013: Донецк: Донецкий национальный технический университет, 2013. - С. 276-282.

11. RFC 7478 - Web Real-Time Communication Use Cases and Requirements Режим доступу до ресурсу: https://tools.ietf.org/html/rfc7478

12. RFC 5245 - Interactive Connectivity Establishment ICE: A Protocol for Network Address Translator NAT Traversal for Offer/Answer Protocols Режим доступу до ресурсу: https://tools.ietf.org/html/rfc5245

13. RFC-4566. – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc4566

14. Forouzan, B.A. TCP/IP Protocol Suite. / B.A. Forouzan / McGraw-Hill, New York, 2002 – 979p.

15. RFC6263 Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, DOI 10.17487/RFC6263, June 2011, – Режим доступу до ресурсу: http://www.rfc-editor.org/info/rfc6263

16. RFC 3489 Rosenberg J. STUN: Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). / J.Rosenberg, J. Weinberger, C. Huitema, R. Mahy. / RFC 3489, IETF, Mar. 2003.

17. RFC5766 Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

18. Mark W., ANSI/IEEE 1471 and systems engineering. / Mark W., Maier, David Emery, Rich Hilliard. / Systems Engineering 7.3— 2004. — 257-270p.

19. Шевчук Р.П. Підвищення ефективності клієнт-серверних систем середньої складності / Р.П. Шевчук., А.І. Яцинич // Вісник Тернопільського державного технічного університету. — 2010. — Том 15. — № 1. — С. 182—186.

20. Gamma E. Design Patterns: Elements of Reusable Object-Oriented Software, /Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides / Addison-Wesley, Reading, MA, USA: Addisson-Wesley, 1995.

21. Berson, A. Client - server architecture. / Berson, Alex. / New York, NY : McGraw-Hill, 1992. - 452 p.

22. Пат. №. 6,996,800 Lucassen, John M., and Stephane H. Maes. "MVC (model-view-controller) based multi-modal authoring tool and development environment." U.S. Patent №. 6,996,800. 7 Feb. 2006

23. Krasner G. E. A description of the model-view-controller user interface paradigm in the smalltalk-80 system / Krasner, Glenn E., Stephen T. Pope //Journal of object oriented programming. – 1988. – Т. 1. – №. 3. – С. 26-49.

24. Mikowski M. Single page web applications: JavaScript end-to-end. / Mikowski M., Powell J. / – Manning Publications Co., 2013. - 432 p.

25. Flanagan, David. JavaScript: the definitive guide. " O'Reilly Media, Inc.", 2006. - 994 p.

26. Пат. № 7509584 США. Moser M. Dynamic ECMAScript class loading : пат. 7509584 США. – 2009.

27. ECMAScript I. S. O. ISO/IEC 16262: 1998 //ECMAScript Language Specification. Available from ECMA (European Computer Manufacturers Association) at http://www. ecma. ch/ecma1/STAND/ECMA-262. HTM.

28. Schmiedehausen K. Single Page Application Architecture with Angular. /Schmiedehausen Kim. /Technology and Communication. 2018. - 44 p.

29. Фаулер М. Шаблоны корпоративных приложений. /М. Фаулер / Пер. с англ. — М.: ООО «ИД Вильямс», 2010. — 544 с.

30. Вязовик Н. А. Программирование на Java / Н.А. Вязовик/ — М.: ИНТУИТ.РУ, 2016. – 604 с.

31. Dziadosz, Radoslaw, et al. Liquibase: Version Control for Database Schema. 2017.

32. Eagle, Mark. Wiring your web application with open source java. 2004-04-07. http://www. onjava. com/pub/a/onjava/2004/04/07/wiringwebapps

33. Пат. № 7,565,443. 21 Rossmanith, Stefan, et al. "Common persistence layer." U.S. Patent № 7,565,443. 21 Jul. 2009

34. Furukawa, Y. Web-based control application using WebSocket. / Y.Furukawa, ICALEPCS, 2011, – 673-675p.

35. RFC 7395 Stout, L., Moffitt, J., Cestari, E. (2014). An extensible messaging and presence protocol (XMPP) subprotocol for websocket (№. RFC 7395) режим доступу: https://www.rfc-editor.org/rfc/pdfrfc/rfc7395.txt.pdf

36. Dahl D. A. The W3C multimodal architecture and interfaces standard / D.A. Dahl //Journal on Multimodal User Interfaces. – 2013. – Т. 7. – №. 3. – С. 171-182.

37. Пат. № 15/364,676 STEPHEN, Gregor Leonard; BIRSE, Stuart. Data migration. U.S. Patent Application № 15/364,676, 2018.

38. Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей [Електронний ресурс] / За заг. ред. О.А. Гавриша. – Київ : НТУУ«КПІ», 2016. – 28с. — Режим доступу: http://foundry.kpi.ua/uk/news/9-novini/764-metodychni-rekomendaciji-do-vykonannja-rozdilu-magisterskyh-dysertacij-dlja-studentiv-inzhenernyh-specialnostej.html

ДОДАТОК А

Клієнт-серверний програмний продукт
для забезпечення відеоконференцій

Акт впровадження
УКР.НТУУ«КПІ ім. Ігоря Сікорського».ТВ3131_18М

Аркушів — 1

Київ – 2018

СКАН АКТА С ИДХ

ДОДАТОК Б

Клієнт-серверний програмний продукт
для забезпечення відеоконференцій

Апробації (конференції)
УКР.НТУУ«КПІ ім. Ігоря Сікорського».ТВ3131_18М

Аркушів — 6

Київ – 2018

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Національний університет “Києво-Могилянська академія”
Вища економіко-гуманітарна школа (Польща)

СТАЛИЙ РОЗВИТОК — XXI СТОЛІТТЯ: УПРАВЛІННЯ, ТЕХНОЛОГІЇ, МОДЕЛІ

Дискусії 2018

Колективна монографія

Київ, Україна
2018

УДК 66.012:658.567.1:368.075.8
ББК 65.9(4УКР)-98

Рекомендовано до друку
друку ІТГІП НАН України (Протокол №12 від 12.12.2018)
та Вченою радою Донецького держ авного університету
управління, м. Маріуполь (Протокол №12 від 14.12.2018)

Рецензенти:

чл.-кор. НАН України, д.т.н., проф. Трофимчук О.М. (Інститут телекомунікацій та глобального інформаційного простору НАН України)
д.е.н., проф. Микитенко В.В. (ДУ "Інститут економіки природокористування та сталого розвитку НАН України")
д.т.н., проф. Черноусенко О.Ю. (НТУУ "Київський політехнічний інститут імені Ігоря Сікорського")
д.г.н., проф. Мезенцев К.В. (Київський національний університет ім. Тараса Шевченка)
д.е.н., с.н.с. Пилипів В.В. (Київський національний економічний університет ім. Вадима Гетьмана)
д.е.н., с.н.с. Чукаєва І.К. (ДУ "Інститут економіки та прогнозування НАН України")

Сталий розвиток — XXI століття: управління, технології, моделі. Дискусії 2018: колективна монографія / Міненко М.А., Бендюг В.І., Комариста Б.М. [та ін.]; НТУУ "Київський політехнічний інститут імені Ігоря Сікорського"; Національний університет "Києво-Могилянська академія"; Вища економіко-гуманітарна школа / за наук. ред. проф. Хлобистова Є.В. — Київ, 2018. — 620 с.

Науковий редактор — доктор економічних наук, професор Хлобистов Є.В.

Збережено авторську орфорграфію, пунктуацію і стилістику.
Відповідальність за зміст матеріалів несуть автори.

Результати дослідж ень, оприлюднені в колективній монографії, були обговорені на V Між народній науково-практичній конференції "Сталий розвиток — XXI століття: управління, технології, моделі (наукові читання імені Ігоря Н едіна)", яка відбулася 23-24 ж овтня 2018 року в м. Києві.

ISBN: 978-83-63649-17-3

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів м. Київ, 24-27 квітня 2018 року,

ТОМ 2

Київ- 2018

УДК 004.62

Магістрант 5 курсу, гр. ТВ-71мп Горбенко О.Ю.
Доцент, к.т.н. Третяк В.А.

ВЕБ-РЕСУРС ДЛЯ ЗАБЕЗПЕЧЕННЯ ПРОВЕДЕННЯ ДИСТАНЦІЙНИХ ЛЕКЦІЙНИХ ЗАНЯТЬ

На сьогоднішній день, інформатизація та глобалізація мають велике значення для забезпечення пошуку інформації. Одним із прикладів даної інформації є забезпечення дистанційного навчання, та, як окремий випадок, забезпечення проведення дистанційних лекційних занять з допомогою Всесвітньої мережі Інтернет.

Наш програмний продукт було спроектовано з використанням кращих практик мікросервісної архітектури, що дає можливість до гнучого розширення програмного продукту, що складається з модулю проведення лекційних занять, бази знань, в якій зберігаються навчальні матеріали та посилання на літературу, модуля кімнат спілкування та модулю автоматичної перевірки теоретичних знань та тестування реалізацій практичних занять.

Мікросервісна архітектура наразі є прровідною практикою для побудови масштабних програмних систем. Основними принципами і перевагами є:

- модульність системи, в якій кожний з компонентів може бути реалізований на різними мовами програмування та базах даних [1];
- простота в реалізації та проектуванні масштабування програмного компоненту та програмної системи загалом і групової розробки;
- простота розгортки в якості розподіленої системи, де кожен компонент може бути розгорнутий фізично на різних машинах.

Було проаналізовано основні концепти та принципи мікросервісної архітектури з використанням гнучкої ахітектури програмного забезпечення.

Нами було спроектовано арзітектура мікросервісів та розроблено такі компоненти:

- Registry + discovery service, що відповідає за балансування екземплярів мікросервісів та пошук екземплярів нових мікросервісів;

Gateway service, що відповідає за маршрутизацію запитів та зберігає всю клієнтську частину програмного продукту.

Було спроектовано та реалізовано мікросервіси для відео зв‘язку та кімнат спілкування в рамках заявленої предметної області.

Таким чином, було проведено аналіз технологій реалізації складних розподілених систем, що характеризуються наявністю великої кількості користувачів та високим навантаженням, в результаті чого обрано мікросервісний шаблон архітектури. Спроектовано основні мікросервіси системи та їх взаємодію.

Перелік посилань:

1. Newman S. Building microservices: designing fine-grained systems. – " O'Reilly Media, Inc.", 2015.